ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

PLATAFORMA WEB PARA LA EXTRACCIÓN DE TRAZAS EN SISTEMAS HPC

(Web platform for trace extraction in HPC systems)

Para acceder al Titulo de

INGENIERO DE TELECOMUNICACIÓN

Autor: Iván Pérez Gallardo

Abril – 2014

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: ***Iván Pérez Gallardo***
Director del PFC: Enrique Vallego Gutiérrez, José Luis Bosque Orero

Título: "Plataforma web para la extracción de trazas en sistemas HPC"
Title: "Web platform to trace extraction in HPC systems"

Presentado a examen el día: 28 de abril de 2014

para acceder al Título de

INGENIERO DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Beivide Palacio, Ramón
Secretario (Apellidos, Nombre): Vallejo Gutiérrez, Enrique
Vocal (Apellidos, Nombre): Basterrechea Verdeja, José

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del PFC
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Proyecto Fin de Carrera Nº
(a asignar por Secretaría)

AGRADECIMIENTOS

Me gustaría que estas palabras sirvan para expresar mi gratitud a aquellas personas que durante estos años han contribuido en la finalización de estos estudios.

A todo el profesorado que de alguna forma ha estado presente en mi proceso de formación.

A todos aquellos compañeros con los que he pasado todos estos años, en especial a David, Álvaro, Isaac, Ángel, Pablo, Iván e Ismail, que han tenido que aguantar durante muchos años mi extravagancia. Aquí incluyo a mis hermanos y primos, como amigos para mi, que también han tenido que sufrir esta peculiar cualidad.

Al grupo de Arquitectura y Electrónica de Computadores por su colaboración en el desarrollo de este proyecto.

A mis directores de proyecto, Enrique y José Luis, por las enseñanzas, los consejos, la amabilidad y la paciencia que me han ofrecido a lo largo del transcurso de este proyecto.

A mi tío José con el que he convivido estos años y del que estaré eternamente agradecido por todo lo que me a ofrecido sin pedirme nada a cambio.

A mis padres por todo el soporte que he recibido de ellos desde que nací, y de los que estoy realmente orgulloso.

A todos ellos, muchas gracias
Iván

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Los sistemas de *High Performance Computing* (HPC) ejecutan programas paralelos que aprovechan la capacidad de múltiples nodos de cómputo del sistema. Para ello, los diferentes procesos de cada nodo se comunican entre sí mediante algún tipo de mecanismo de comunicación, típicamente mensajes MPI. La optimización de las aplicaciones para estos sistemas precisa de mecanismos que informen de los eventos que ocurren en el sistema, permitiendo analizar las diferentes fases de ejecución, las comunicaciones entre nodos, impacto en la escalabilidad, etc. La obtención de trazas de la ejecución de las aplicaciones permite visualizar estos parámetros. Además, las trazas de aplicaciones paralelas son muy útiles en la investigación en el entorno de HPC, típicamente empleándose en herramientas de simulación.

La extracción de trazas resulta en muchas ocasiones una tarea compleja, por la dificultad de acceso a sistemas de HPC lo suficientemente grandes, por la complejidad de uso de las herramientas de extracción o por lo tedioso del proceso. Por estos motivos, las trazas de aplicaciones se consideran muy valiosas en el entorno académico, y en ocasiones no se comparten, impidiendo la repetibilidad de los experimentos.

El objetivo de este proyecto ha sido desarrollar una plataforma web capaz de extraer trazas de programas paralelos MPI en sistemas HPC y compartirlas abiertamente a modo de repositorio. La funcionalidad de la plataforma desarrollada permitirá a los usuarios extraer trazas de sus propios programas MPI así como de programas que hayan sido publicados por otros usuarios, ofreciéndoles una interfaz intuitiva, capaz de abstraerlos de la complejidad de estos sistemas. Además, el sistema asigna automáticamente una licencia de uso a las trazas generadas que permite la atribución de autoría, e incluye un repositorio compuesto por las trazas de los usuarios que permitirá comparar y analizar el comportamiento de estos programas en distintas configuraciones.

El desarrollo del proyecto ha comprendido el análisis y elección de un *framework* de desarrollo web como base para la parte del servidor; el aprendizaje del funcionamiento de sistemas de colas en entornos HPC, así como de la herramienta Extrae de extracción de trazas; y finalmente la validación del sistema mediante varios casos de uso. El sistema ha sido desarrollado sobre Altamira, un sistema HPC de la Universidad de Cantabria, y aun así el diseño es suficientemente flexible como para añadir nuevos sistemas cuando estén disponibles. Se ha comprobado que, utilizando la herramienta diseñada, es posible obtener trazas sin precisar de un conocimiento detallado de la arquitectura del sistema, así como acceder fácilmente a las trazas obtenidas por otros usuarios.

ABSTRACT

High Performance Computing (HPC) systems take advantage of the capacity of multiple compute nodes of the system to execute parallel programs. The different processes on the system nodes communicate between them using some kind of communication method, typically MPI messages. Optimizing applications for these systems requires some method to report the events that happen in the system, which allows analyzing the different stages of the execution, the communication between nodes, the impact in the scalability, etc. The extraction of traces of the execution of the applications allows visualizing these parameters. Additionally, such traces of parallel applications are very useful for research and development of HPC systems, typically used as the inputs of simulation tools.

The trace extraction process is a hard and tedious task, due to the difficulty of accessing large-enough HPC systems, the complexity of the extraction tools or the multiple steps in the process. For these reasons, traces of applications are typically considered very valuable in the academic environment, and very frequently they are not shared openly, impeding the repeatability of the experiments.

The goal of this project is to develop a web platform capable of automating the trace extraction process of MPI parallel programs in HPC systems. The functionality implemented in the platform allows users to run their own MPI programs or other programs available in the system to extract traces, providing an intuitive interface, which abstracts them from the complexity of the HPC systems. Moreover, the platform assigns automatically a license to the generated traces, allowing for the attribution to the author, and includes an open repository composed by the users's traces, allowing for the comparison and analysis of the behaviour of these programs in different configurations.

The development of the project has involved the analysis and selection of a web development framework as a basis of part of the server; the learning of the operation of queue systems in HPC environments, and the trace extraction tool called Extrae; and finally the validation of some cases of use of the design system. The system has been developed using Altamira, an HPC system in the University of Cantabria, but the design is flexible enough to be easily extended to other clusters whenever they are available. The developed tool has proven useful to acquire traces of parallel applications without requiring a deep knowledge of the system architecture.

CAPÍTULO 1

INTRODUCCIÓN

La computación de alto rendimiento o *High-Performance Computing* (HPC), es el uso de computadores paralelos o *clusters* para resolución de problemas científicos, de ingeniería o empresariales con un alto grado de complejidad. La composición de estos *clusters* puede variar desde estar formados únicamente por una gran instalación con elementos homogéneos de gran capacidad computacional, a multitud de sistemas heterogéneos conectados entre sí. El término HPC también puede hacer referencia a *High-Performance Computer*, es decir sistemas de computo de alto rendimiento, por lo que a lo largo de este documento se hará uso de ambas definiciones que serán claramente diferenciables según el contexto en el que se ubique.

Habitualmente las aplicaciones ejecutadas en estos sistemas son soluciones a problemas de los que no es posible deducir soluciones analíticas exactas y para los que es necesario recurrir a modelos computacionales con los que obtener resultados de gran precisión. Este tipo de aplicaciones requiere de una gran capacidad computacional y es por ello que se ejecutan sobre estos sistemas de alto rendimiento con los que es posible generar esos resultados en un plazo de tiempo relativamente corto. Como ejemplos de estos sistemas se pueden nombrar a los famosos supercomputadores de la compañia Cray [3], los modelos de la serie Blue Gene [33] de IBM, el supercomupator MareNostrum diseñado por IBM y ubicado en el Centro Nacional de Supercomputación (*Barcelona Supercomuping Center*, BSC) siendo el sistema con la mayor capacidad de España según el *ranking* de Top500 [29], y el supercomputador Altamira ubicado en la Universidad de Cantabria (UC) y que al igual que MareNostrum, forma parte de la Red Española de Supercomputación (RES).

Estos *clusters* están compuestos por los elementos de un computador convencional, procesador, memoria, sistema de almacenamiento, sistema operativo, etc, pero como es de suponer en mayor cantidad. Estos sistemas están formados por nodos de computo enlazados entre sí, permitiendo la ejecución de las aplicaciones implementadas con mecanismos de programación paralela. Cada uno de estos nodos tiene uno o varios procesadores así como memoria y su propio sistema de almacenamiento, y no necesariamente han de ser iguales.

Un aspecto fundamental en el rendimiento de estos sistemas son sus redes de interconexión. Estas redes están compuestas por las NIC (*Network Interface Card* o adapatador de red) de los nodos, conmutadores y enlaces, y se caracterizan por su topología, su mecanismo de control del tráfico, el enrutado y la técnica de conmutación, entre otros. Para evaluar el comportamiento de una aplicación en ejecución en una determinada configuración de red se hace uso de herramientas capaces de capturar los eventos que surgen en los distintos procesos de la ejecución.

Estas capturas reciben el nombre de **trazas** y tienen un gran valor en el estudio y diseño de estas aplicaciones. Existen diversos tipos de trazas, algunas se centran en el comportamiento de las CPU de los nodos de computo, otras en el acceso a los distintos niveles de memoria, otras en las técnicas de comunicación entre los procesos de una aplicatión, etc.

Es importante destacar el valor de las trazas desde el punto de vista de la investigación. Esta información permite caracterizar el comportamiento de la aplicación, simular el rendimiento de la aplicación en un sistema alternativo, valorar el sistema en el que se ejecuto la aplicación, por dar algunos ejemplos. Además desde el punto de vista del desarrollador de aplicaciones estas trazas son una fuente de información con la que poder mejorar el rendimiento de las aplicaciones ajustándose a las configuraciones de estos sistemas.

1.1 Motivación

El uso de estos sistemas de alto rendimiento y de las herramientas necesarias para la captura de trazas requiere de un aprendizaje, que en ocasiones supone un gran esfuerzo y una gran cantidad de tiempo sino se esta familiarizado con una aplicación o sistema en particular. Además la duración de la ejecución de estas aplicaciones puede variar desde unos pocos minutos a varios días.

Estas trazas pueden centrarse en abordar distintos campos, desde capturar la sucesión de eventos relacionados con la ejecución de las instrucciones que forman la aplicación en la CPU, pasando por el registro de los eventos relaconados con la memoria como pueden ser las instrucciones de acceso a memoria “load/store” o los fallos que se producen en los distintos niveles de la jerarquía de memoria, hasta llegar a registrar la información referente a la comunicación de red.

La motivación de este proyecto reside en crear una herramienta con la que sus usuarios puedan lanzar aplicaciones para la captura de trazas sin necesidad de tener acceso directo a estos sistemas de alto rendimiento. El tipo de traza a capturar se enfocará en la información referida a la comunicación de red, en la que se emplean modelos de transferencia de mensajes como MPI o modelos de memoria compartida como Pthread (POSIX *Threads*). Además se pretende almacenar estas trazas para generar un repositorio público al que cualquier investigador pueda acceder. Una herramienta con la que los usuarios eviten la tediosa tarea de monitorizar el estado de la ejecución y que de forma automática registre las trazas cuando estas han sido completadas. Con ello se pretende, por una parte, facilitar y acelerar el proceso de captura de trazas, y por otra parte, ofrecer a la comunidad científica de este ámbito la posibilidad de compartir este tipo de información para su estudio y manipulación.

La idea de disponer de una colección de trazas de características diversas, o similares en las que se haya empleando distintas configuraciones o lanzadas en distintos *clusters*, puede suponer una gran ayuda tanto en la optimización de aplicaciones paralelas, cómo en el diseño y evaluación de *clusters* mediante simulación, que actualmente esta supeditado a la búsqueda de sistemas de mayor rendimiento y menor consumo energético. Por ejemplo el uso de estas trazas para evaluar el comportamiento de una aplicación en un sistema puede proporcionar a los desarrolladores de dicha aplicación un medio con el que determinar qué aspectos, o qué características pueden mejorar para lograr un rendimiento mayor. Un claro ejemplo de uso de estas trazas para el diseño de sistemas se encuentra en el desarrollo de herramientas de simulación de redes, dado que estas trazas proporcionan a los desarrolladores una gran cantidad de información

sobre el comportamiento de los sistemas que han de emular.

En definitiva se trata de automatizar el proceso de obtención de la traza y ponerla en un repositorio público; a la vez hay que asegurar el acceso no malintencionado a los recursos y proteger las aplicaciones.

1.2 Problema

La problemática de esta idea yace fundamentalmente en dos factores. La intención de ofrecer a los usuarios acceso a sistemas ajenos implica la necesidad de algún método para evadir usos malintencionados de ella. El otro factor reside en la generación de datos, las trazas, en procesos de investigación y disponibles para otros investigadores por medio del repositorio público. La intención de este repositorio es fomentar la distribución y manipulación de estos datos por ello se hace necesario el uso de licencias con las que proteger legalmente esta información.

Ligado a la información proporcionada por las trazas de una aplicación se encuentra su código fuente, que en muchas ocasiones puede proporcionar información complementaria. Dado que se pueden tratar de aplicaciones desarrolladas por el propio usuario, o de aplicaciones que tienen condiciones especiales de uso es importante que la herramienta disponga de algún sistema que permita al usuario proteger estas aplicaciones.

1.3 Objetivos

Para poder ofrecer estos servicios se ha propuesto el diseño de una plataforma web, totalmente independiente de estos sistemas de alto rendimiento, en la que los usuarios puedan subir códigos fuente de aplicaciones, y que se encargará de forma automática de configurar y utilizar las herramientas de extracción de trazas en los *cluster*, para posteriormente descargar las trazas generadas e incluirlas en la base de datos de la plataforma.

Con la motivación y la problemática presentes, los objetivos principales de este proyecto se pueden resumir en: la creación de una herramienta que facilite a sus usuarios la captura de trazas de la ejecución de programas paralelos en sistemas HPC ajenos; la creación de un repositorio de trazas en el que se fomente la compartición de estas trazas de gran valor entre los investigadores de este área, evitando la repetibilidad de los experimentos. Estos dos objetivos se concretan en los siguientes puntos:

- Crear un servicio web capaz de automatizar el proceso de generación de trazas, que consiste en la ejecución de la aplicación haciendo uso de las herramientas de extracción de trazas en un sistema de computación de alto rendimiento y su posterior registro en este servidor web.
- Realizar una interfaz web que permita abstraer, en la medida de lo posible, de la complejidad de las herramientas necesarias para la extracción de trazas.
- Diseñar una arquitectura flexible, que permita añadir y disponer de distintos sistemas de computación de alto rendimiento.
- Generar un repositorio con las trazas almacenadas.

- Proporcionar al usuario un sistema de protección con el que bloquear el acceso al código fuente de sus aplicaciones.
- Controlar el acceso de los usuarios que deseen hacer uso de los servicios que impliquen acceder de forma indirecta a los sistemas de computación de alto rendimiento.
- Disponer de un sistema de protección de la información generada por la herramienta en base a la elección de una licencia, que permita al usuario definir las condiciones de uso de las trazas.
- Permitir a los usuarios enviar sus propias trazas.

1.4 Estructura del Documento

Este documento se estructura en seis capítulo.

El primer capítulo lo compone esta introducción, en la que se establece la temática del proyecto, su motivación, su problemática y sus objetivos.

En el capítulo 2, se estudiarán las herramientas y sistemas empleados para el desarrollo del proyecto. Se explicará que es Drupal y por qué se ha empleado. Se introducirá al lector la herramienta de captura de trazas, Extrae, analizando de forma superficial su manejo. Por último se hablará del sistema gestor de colas de los *cluster*, que como se verá es el medio por el que el usuario puede interactuar con las unidades de computo de los sistemas HPC.

En el capítulo 3 se examinará profundamente el proceso de diseño de la herramienta. Se indicarán los requisitos establecidos que ha de cumplir la aplicación, los casos de uso, y el problema de codiseño entre el *cluster* utilizado y el servidor web.

En el capítulo 4 se analizará el proceso de implementación de la plataforma web y las pautas que se han de seguir para incluir nuevos sistemas de computo a la plataforma web.

El capítulo 5 recoge un ejemplo con el que se verificará el correcto funcionamiento de la plataforma web.

Por último en el capítulo 6 se determinarán las conclusiones obtenidas al finalizar el proyecto. Se indicarán los objetivos cumplidos, así como posibles futuras mejoras.

CAPÍTULO 2

TECNOLOGÍAS Y HERRAMIENTAS

En este capítulo se estudiarán las herramientas empleadas en el desarrollo de esta aplicación web: qué son, cómo se usan, y qué conceptos son necesarios conocer de ellas para comprender y desarrollar la arquitectura de la plataforma web objeto de este proyecto.

2.1 Servidor web, Drupal

El servidor web es el medio que tendrán los usuarios para lanzar y capturar la ejecución de aplicaciones en los sistemas HPC. Es decir, será la interfaz entre los usuarios y los *clusters*.

Para la creación de este servicio se plantearon dos posibilidades. La primera consistía en optar por realizar una plataforma web completamente personalizada utilizando herramientas básicas de programación como HTML, PHP, CSS y un gestor de bases de datos, como medios para mostrar el contenido general, implementar la funcionalidad, el estilo, y el almacenamiento de la información respectivamente. La otra opción era emplear un *framework* como base sobre la que añadir la funcionalidad necesaria para lograr los resultados deseados. Dentro de esta opción existen varias alternativas como Drupal [12], Wordpress [30], Joomla [7], Magento [8], etc. De entre todos ellos, se optó por Drupal ya que con anterioridad ha sido empleado en el grupo de trabajo, y se contaba con experiencia en el desarrollo de aplicaciones similares.

El uso de un *framework*, en concreto Drupal, trae consigo una serie de ventajas. A continuación se listan las más relevantes.

- Abstracción de subsistemas como la gestión de la base de datos, la generación de formularios, la maquetación, etc.
- Facilidad para crear distintos tipos de páginas con contenido.
- Incluye gestión de usuarios.
- Existe un amplio soporte por parte de la comunidad. Esto incluye funcionalidad muy diversa ya creada y fácilmente agregable al sitio web.
- Cuenta con un menú de administración, en el que es posible agregar/modificar permisos de usuario, añadir funcionalidad, cambiar temas visuales, etc, de manera sencilla. Además de

esto cuenta con una aplicación llamada Drush [4] que facilita estas funciones desde linea de comandos, orientada a usuarios más avanzados.

- Requiere de menor tiempo de desarrollo.

Por contra también tiene una serie de inconvenientes:

- Es necesario conocer la herramienta, es decir, entender cómo funciona, comprender su estructura, conocer su API, etc.
- Si el rendimiento es un requisito imprescindible, emplear Drupal puede ser una opción arriesgada dado que es muy probable que implemente más funcionalidad de la necesaria por la aplicación. Además la orientación de su diseño, como se verá posteriormente, es hacer de Drupal una herramienta flexible que permita ser usada en temas dispares lo que puede conllevar a que su optimización en una temática concreta no sea la ideal.

Por estas ventajas, principalmente por el menor tiempo de producción, se ha escogido Drupal como herramienta de trabajo. Además como se verá en el apartado 4.1, Drupal cubre una gran parte de las necesidades de la plataforma web deseada, sin necesidad de añadir nueva funcionalidad.

2.1.1 Drupal

A Drupal se le atribuyen dos definiciones, *Content Management System* (CMS) y *Content Management Framework* (CMF). La primera hace alusión a Drupal como herramienta *software* para gestionar el contenido de un sitio web, como puede ser el contenido de un foro, una *wiki* o un *blog*. La segunda hace referencia a la instrumentación que ofrece a los desarrolladores para modificar o añadir funcionalidad a la propia de Drupal.

Drupal es *Free Open Source Software* (FOSS), es decir, es gratuito y de código abierto, bajo la licencia *GNU Generic Public License* (GNU/GPL). Esta escrito en PHP y se caracteriza por ser un sistema dinámico, almacenando el contenido de la página web en su correspondiente base de datos. Es compatible con distintos gestores de bases de datos, como por ejemplo MySQL, PostgreSQL o SQLite, distintos servidores web, como Apache, Nginx, o Microsoft IIS, y distintos sistemas operativos, como Linux, Mac OSX y Windows.

2.1.2 Estructura de Drupal

Según la documentación oficial de Drupal [14], su estructura esta formada por cinco capas como se ilustra en la figura 2.1 que ha sido obtenida de [12].

1. La más baja, la **capa de datos**, se corresponde con el conjunto de datos, es decir el contenido y la información que Drupal necesita para mostrar las distintas páginas del sitio web. Un concepto muy importante asociado a esta capa, y que será empleado a lo largo de este documento, es el concepto de “nodo”. Los nodos son el contenido principal de Drupal, la información que se representa en las distintas páginas de Drupal, al margen de los menús y los bloques que pueden incluir contenido adicional como se estudiará posteriormente.

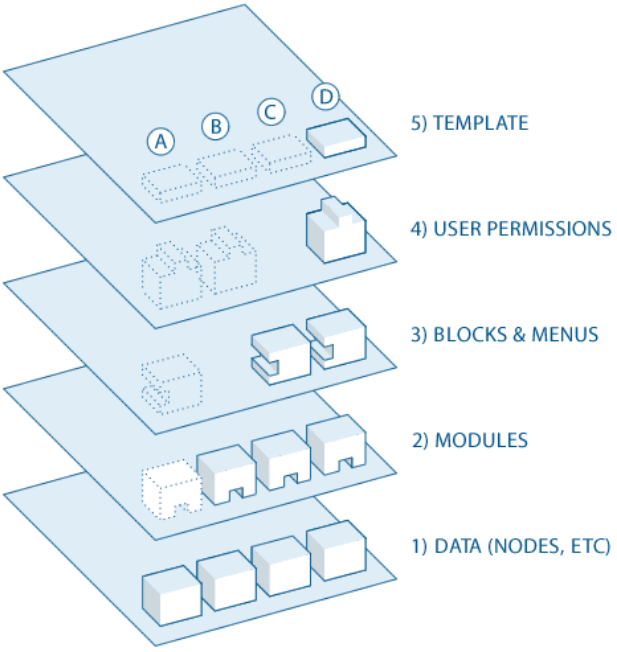


Figura 2.1: Estructura de Drupal.

Pueden existir distintos tipos de nodos caracterizados por una serie de campos asociados a ellos, pudiéndose crear tantos nodos de un tipo como sea necesario.

2. En la segunda capa se sitúan los **módulos**. Los módulos son la principal característica de Drupal, ya que son sus elementos funcionales. Gracias a estos elementos se puede decir que la estructura de Drupal es modular y flexible, ya que añadiendo o quitando estos módulos el funcionamiento del sistema será distinto. Los módulos se pueden dividir en tres categorías:

   - Módulos del núcleo. Son los incluidos por defecto en Drupal, y forman parte de sus funciones básicas. Algunos de ellos se pueden habilitar y deshabilitar, siendo posible ajustar el rendimiento y funcionalidad según las necesidades de la aplicación.
   - Módulos aportados por la comunidad. Son módulos de acceso libre, publicados fundamentalmente en el sitio web principal de Drupal.
   - Módulos personalizados. Hacen referencia a los módulos creados por usuarios de Drupal. La creación de estos módulos requiere de conocimientos de Drupal como CMF, es decir, conocer la estructura interna de Drupal, programación PHP, la API de Drupal, y en ocasiones HTML5 o XHTML, CSS y JavaScript.

3. La tercera capa esta formada por los **bloques y los menús**. Los bloques son entidades en las que se puede representar información. Únicamente cierto contenido que haya sido especificado en algún modulo podrá ser representado y se podrá colocar en distintos lugares de la página, los cuales están definidas en las plantillas correspondientes a la quinta capa. Por otro lado, los menús proporcionan la navegación entre las distintas páginas del sitio web. En estos menús se define la ruta al contenido de Drupal.

4. En la cuarta capa se sitúan los **permisos de usuario**. En esta capa es donde se configuran y

determinan las acciones que cada usuario es capaz de realizar. Los permisos son asignados a roles, y los usuarios son categorizados dentro de estos roles.

5. En la última, la quinta, se encuentran las **plantillas**. Esta capa hace referencia al formato y estilo de visualización de Drupal. Estas están construidas principalmente en XHTML o HTML5, CSS con algunas variables intermedias en PHP, y en ellas se establece en qué lugares de la página web puede ir colocado el contenido de Drupal, ya sean bloques, menús, el contenido principal, etc. Estas plantillas junto con otro tipo de archivos e imágenes compondrán los temas, pudiéndose tener varios instalados en el sistema y seleccionar uno, e incluso asignar diferentes temas para diferentes tipos de usuario.

El flujo de ejecución de Drupal va desde las capas inferiores a las superiores. De esta forma si se quiere añadir una nueva funcionalidad, ésta ha de ser agregada en todas las capas que se requiera. Esto es lo que se representa en la figura 2.1 con los casos A, B, C y D. En el caso A, se representa un módulo instalado en el sistema, pero que no ha sido habilitado. De esta forma la funcionalidad proporcionada no afectará a Drupal. En el caso B y C, se muestran dos módulos cargados y habilitados. Se aprecia que falta una pieza en la capa de menús y bloques para el caso B, y otra pieza en la capa de permisos de usuario para el caso C, mientras que las piezas de las capas superiores están trazadas en lineas discontinuas. Con estos casos se quiere indicar que el funcionamiento de un módulo puede no ser el esperado debido a la configuración de las capas superiores al módulo. De esta forma en el caso B, no se visualiza la salida del módulo, ya sea en un menú o en un bloque, puesto que no se ha configurado una entrada en el menú, no se ha habilitado o especificado la posición del bloque definido en el módulo, o cualquier problema similar. Por otra parte en el caso C, la no visualización de la utilidad del módulo esta causada por la configuración de los permisos de usuario, no dando permisos al usuario del empleo de la funcionalidad proporcionada por este módulo. Por último en el caso D, se muestran todas las piezas de las distintas capas en trazo continua, indicando que las condiciones en cada una de las capas son las adecuadas para que el usuario final pueda visualizar la salida proporcionada por el módulo.

2.1.3 API de Drupal

Drupal proporciona una API [17] con la que los desarrolladores puedan crear sus propios módulos. A continuación se describen las secciones de esta API con mayor relevancia en el diseño del sistema.

La sección más importante es la que abarca los *hooks*, ya que son el medio de interacción de los módulos con el núcleo de Drupal. Un *hook* es una función PHP con un nombre que sigue la siguiente sintaxis: [nombre del módulo]_[nombre del hook](), donde “nombre del modulo” se corresponde con el nombre del módulo donde esta localizada la implementación de esta función, y “nombre del hook” con el nombre que recibe el *hook*. De esta forma en la documentación de Drupal se listan los *hooks* disponibles todos siguiendo el mismo patrón. “hook_node_insert”, “hook_node_save”, “hook_form_alter”, son algunos ejemplos. Si se quiere utilizar un *hook* en un módulo, basta con sustituir “hook” por el nombre del módulo. Por ejemplo en el caso de que el nombre del módulo sea “trace_state” y que se desee utilizar el *hook_node_insert* bastaría con definir la función dentro del código del módulo usando el nombre “trace_state_node_insert”.

Existen multitud de *hooks*, cada uno de ellos con sus propios parámetros de entrada y salida, que atienden a distintos eventos o estrictamente hablando a *callbacks* del sistema. Todas las

funciones que implementen este *hook* irán siendo ejecutadas, sin interferirse entre sí. De esta forma, cuando un evento asociado a un *hook* ocurre en el núcleo de Drupal, este, invoca las implementaciones del *hook*, de los distintos módulos habilitados en el sistema, añadiendo la funcionalidad correspondiente.

Por ejemplo, "hook_node_insert" responde a la creación de un nodo, concretamente es invocado cuando el nodo esta preparado para ser registrado en la base de datos. Cuando un usuario envía la información solicitada en un formulario para la creación de un tipo de contenido, Drupal invocará a las funciones que implementen este *hook*. En el caso de este proyecto, este *hook* se ha utilizado para poder procesar los datos introducidos por el usuario y poderlos enviar al *cluster* para la ejecución de una aplicación y llevar a cabo la captura de trazas, como se verá en el apartado 4.1.5.

Otra sección de la API de Drupal relevante es la correspondiente a la base de datos. Por medio de esta API, es posible manejar la información almacenada en la base de datos del sitio web sin importar el servidor de bases de datos sobre el que se haya realizado la instalación. De esta forma, independientemente de la configuración de Drupal, los módulos serán compatibles.

También se han de mencionar tres secciones más que se han empleado en el desarrollo de esta aplicación.

- La API referente a la creación o modificación de formularios, que permite simplificar el código, y tener una mayor consistencia en su procesado y representación.
- El sistema de gestión de ficheros. Gracias a esta API se facilita el manejo de ficheros en Drupal por medio de una serie de funciones y un nuevo tipo de objeto denominado "file". De esta forma registrar nuevos ficheros en la base de datos, eliminarlos, cambiar los permisos de uso, etc, es simple e intuitivo.
- El sistema de menús que permite definir el sistema de navegación desde el punto de vista del usuario y el sistema de direccionamiento encargado de responder ante la solicitud de una dirección concreta.

2.2 Altamira

Altamira es un supercomputador ubicado en la Universidad de Cantabria con una capacidad de computo de 80 Teraflops y que está compuesto por 240 nodos IBM idataplex dx360m4, cada uno de ellos con doce núcleos Intel Xeon E5-2600 v2. Tiene una red de interconexión de los nodos Infiniband FDR10 que cuenta con un ancho de banda de 40 Gbps. Además cuenta con una capacidad de almacenamiento superior a 2 PB.

Además de este *cluster*, al comienzo del proyecto, durante la fase de aprendizaje y familiarización de las herramientas y tecnologías, se trabajó con otro *cluster* llamado Calderón ubicado en la Universidad de Cantabria. Este *cluster* esta formado por alrededor de 180 nodos de computo de distintas características componiendo un sistema heterogéneo a diferencia de la homogeneidad presente en Altamira.

Dado que el objetivo de este proyecto es crear una herramienta flexible a la que poder agregar distintos *cluster* sobre los que realizar la extracción de trazas, en un principio se pensó en trabajar sobre ambos sistemas, pero una vez se analizo el tiempo requerido para incluir ambos sistemas

se determinó que era necesario escoger uno. En este contexto, la facilidad de manejo así como las pruebas previas de extracción de trazas satisfactorias que se realizaron sobre el *cluster* Altamira, hicieron de este sistema el seleccionado.

Como se observará en el apartado 3 este sistema presenta un gran inconveniente que influirá en la implementación de la plataforma. Este inconveniente consiste en que presenta una comunicación unidireccional, es decir solo se podrán establecer conexiones desde equipos externos al *cluster*, bloqueando las conexiones desde el *cluster* a equipos externos. Calderón, en cambio, sí permite establecer este tipo de conexiones, pero dado que la idea de este proyecto es definir un diseño que permita incluir cualquier tipo de *cluster*, la opción más restrictiva, Altamira, será la opción que mejor se ajusta.

Los usuarios de este sistema podrán hacer uso de los distintos nodos de computo para lanzar sus aplicaciones paralelas. El lanzamiento de estas aplicaciones se podrá realizar especificando el número de procesos y nodos a utilizar, pudiéndose indicar desde la ejecución de cada proceso de la aplicación en un nodo distinto, hasta la ejecución de todos los procesos en un mismo nodo, siempre y cuando el nodo disponga del mismo número de núcleos o más que procesos tenga la aplicación.

2.3 Extrae

Extrae [26] es una herramienta software de libre acceso desarrollada por el BSC [25], *Barcelona Supercomputing Center*, que permite la generación de trazas en aplicaciones MPI .

MPI [5] es un estándar de programación paralela basado en el envió de mensajes y que es utilizado sobre todo en sistemas de computación de alto rendimiento o *High Performance Computing* (HPC).

Por otra parte las trazas son ficheros donde se almacena la información referida al comportamiento de una aplicación cuando se ejecuta. En las trazas de los sistemas HPC se captura principalmente información sobre las invocaciones a funciones de comunicación entre procesos, la duración de los estados en cada uno de los procesos, la cantidad de recursos empleados, etc, de la aplicación. Esta información tiene gran valor como método de desarrollo y depuración de dichas aplicaciones, pero aún más en el campo de análisis, diseño y configuración de arquitecturas de red, en los que se usa la simulación.

Particularizando, Extrae captura las invocaciones a funciones de distintos modelos de programación paralela. Modelos de memoria compartida como OpenMP y pthreads, y modelos de transferencia de mensajes como MPI. Al utilizar Extrae se genera un fichero por proceso de ejecución con extensión MPIT. Posteriormente estos ficheros de trazas pueden ser utilizados para realizar simulaciones con la herramienta Dimemas [24] y se pueden unir para formar un único fichero con formato PRV que puede ser visualizado usando la herramienta Paraver [27], ambas también creadas por el BSC . En la figura 2.2 se representa un ejemplo de traza de una aplicación MPI, y en la que se aprecian diferentes estados de los procesos de la aplicación. En azul se representa el estado de ejecución, en azul cian la espera, y en amarillo la relación envió-recepción de mensajes entre procesos.

Para hacer uso de Extrae es necesario enlazar las bibliotecas correspondientes de esta herramienta al programa a ejecutar. Esto se puede llevar a cabo de dos formas, incluyendo las rutinas proporcionadas por la API de Extrae en el código fuente del programa de interés, o bien inyec-

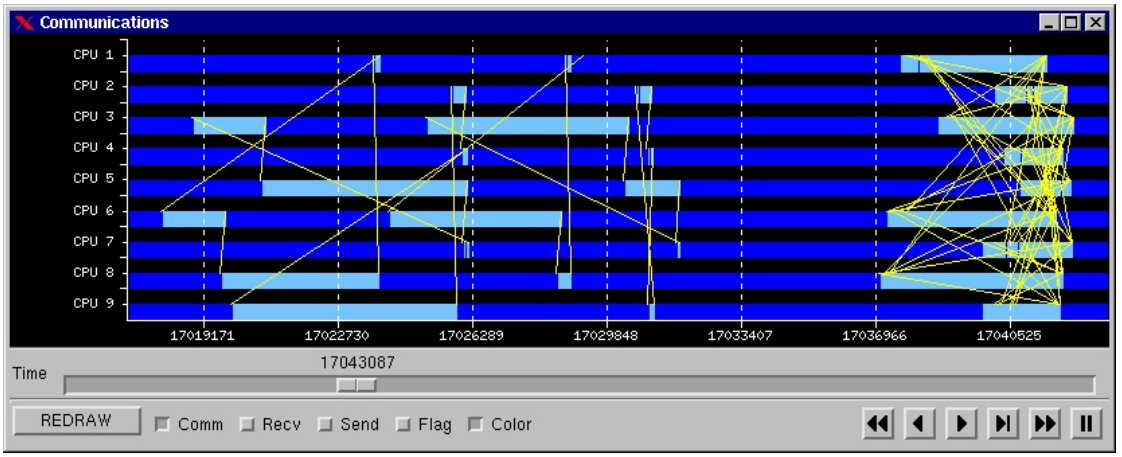


Figura 2.2: Ejemplo de visualización de trazas de una aplicación MPI con la herramienta Paraver. Imagen obtenida de la página web del BSC [25]

tarlas de forma dinámica cuando la aplicación se encuentra en ejecución indicando a Extrae que bibliotecas de comunicación se emplean en la implementación de dicha aplicación.

Extrae es compatible con determinadas herramientas, como *Performance Aplication Programming Interface* (PAPI) [21] o Dyninst [20]. Por ejemplo, PAPI permite hace uso de los contadores *hardware* de los que disponen la mayoría de microprocesadores para recoger información adicional en las ejecuciones. Mientras que la herramienta Dyninst es empleada por Extrae para inyectar de forma dinámica las instrucciones adicionales necesarias para registrar la información durante la ejecución automáticamente, sin necesidad de indicar a Extrae que bibliotecas ha de usar. Estas herramientas son asociadas a Extrae durante su instalación, por lo que no será posible su uso en todas las configuraciones.

Como usuarios de esta aplicación el aspecto más importante de esta herramienta recae en su configuración. Esta configuración permite seleccionar el tipo de información a registrar en las trazas. Para ello es necesario la edición de un fichero XML detallando las distintas opciones configurables de Extrae. Estas opciones siguen la sintaxis del fragmento de código XML 2.1 que se corresponde a una sección del fichero de configuración. Este fragmento se corresponde con la información obtenida de la pila de invocaciones. En él se observa que hay 3 etiquetas, “callers”, “mpi” y “sampling” seguidas de “enabled”, indicando que estas opciones pueden ser habilitadas o deshabilitadas con el uso de “yes” o “no” respectivamente. Además algunas opciones requieren de algún dato adicional para su configuración como es el caso de las opciones “mpi” y “sampling” del ejemplo, que están seguidas de un valor numérico, el cual se corresponde al nivel de profundidad máximo a capturar en invocaciones sucesivas.

```
<callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <sampling enabled="yes">1-5</sampling>
</callers>
```

Código 2.1: Fragmento del fichero de configuración XML de Extrae

Existen multitud de opciones configurables y todas ellas se recogen en el manual de Extrae [32]. Algunas de las secciones con las que cuenta este fichero de configuración son: Trace, referente a información general de la traza; MPI, referido a la captura de la información relacionada con el uso de la biblioteca “MPI” para la comunicación entre procesos mediante el traspaso de mensajes; Callers, asociado a las direcciones de memoria de las rutinas de los distin-

tos procesos de la aplicación y que es usado para enlazar los ficheros de trazas al código fuente de la aplicación; pthread, referido a la recogida de información relacionada con la biblioteca "pthread" usada en la comunicación entre procesos según el modelo de memoria compartida; Counters, ligado al uso de los contadores *hardware* del sistema para la obtención de información en la captura de las trazas; Burst, relacionado con la obtención de información sobre las ráfagas computacionales.

2.4 Gestor de Colas

Un gestor de colas es el *software* encargado de asignar los recursos de un sistema HPC, a los distintos procesos. Este gestor tiene una gran importancia dentro del sistema, ya que decide la asignación de procesos a recursos del sistema a lo largo del tiempo. En la figura 2.3 se representa un ejemplo de una posible estructura de un gestor de colas. En ella se divide al gestor de colas en dos subsistemas, el gestor de recursos y el planificador. El gestor de recursos recopilará información del estado actual de los recursos del sistema. Por otra parte, el planificador se encargará de decidir qué procesos y en qué momento podrán utilizar los recursos en función de la información obtenida por el gestor de recursos. También se representan diferentes colas, algunas accesibles a los usuarios con las que podrán ejecutar, depurar, compilar, etc, trabajos mediante una interfaz. A otras solo tendrá acceso el planificador, como pueden ser las colas de espera a ciertos recursos, en las que este irá colocando los distintos procesos que requieran de dicho recurso. Existen diferentes criterios de organización de las distintas tareas dentro de las colas, siendo el más habitual el basado en prioridades. Este criterio consiste en ordenar las tareas según la prioridad que le otorgue el sistema por uno o varias propiedades como pueden ser, el rol del usuario, el tiempo estimado de ejecución, la cantidad de recursos que necesita, etc. En los sistemas de computo de alto rendimiento, estos recursos principalmente son los distintos nodos del *cluster* y los sistemas de almacenamiento. Además el gestor de colas proporciona una interfaz a los usuarios con la que enviar procesos a las distintas colas del sistema, así como su monitorización, y permiten a los administradores definir nuevas colas o modificar las propiedades de las ya existentes.

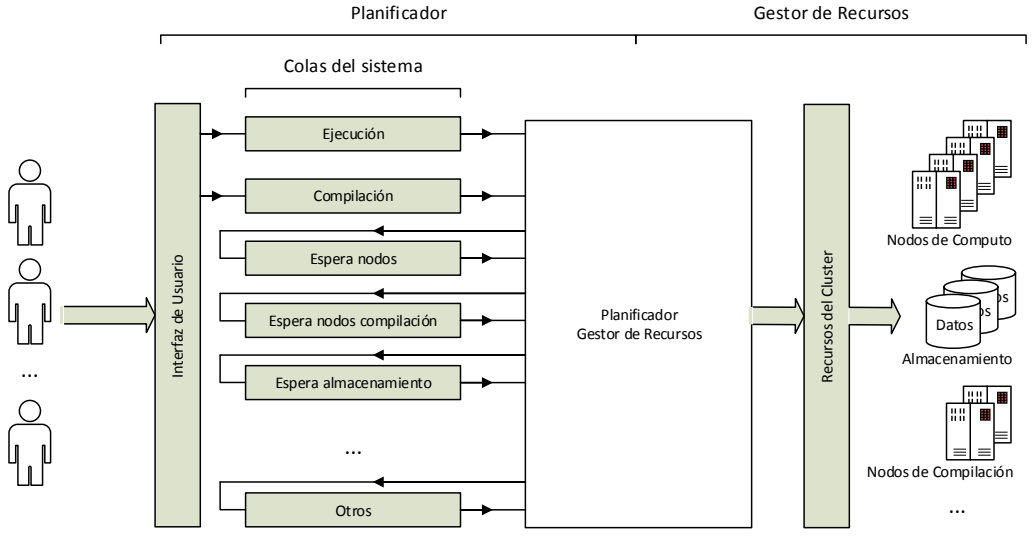


Figura 2.3: Sistema gestor de colas.

El gestor de colas que incorpora Altamira recibe el nombre de *Simple Linux Utility for Resource Management* SLURM [19]. SLURM es un sistema gestor de colas de código fuente abierto, tolerante a errores, y con gran escalabilidad. No requiere de modificaciones del *kernel* y es autocontenido. Como gestor de colas tiene tres funcionalidades.

- Gestión de cuentas de usuario.
- Planificador, proporcionando un entorno de trabajo con el que enviar trabajos a las colas del sistema para ejecutar, depurar, y monitorizar las aplicaciones.
- Gestor de recursos, regulando el uso de los recursos mediante la gestión de colas de trabajos pendientes.

Altamira dispone de una interfaz que envuelve a SLURM creada por el BSC, que permite abstraer al usuario de algunos de los aspectos de este gestor de colas. Esta interfaz esta compuesta por tres comandos que permiten el lanzamiento (mnsubmit), la monitorización (mnq) y la cancelación (mncancel) de trabajos. El comando de lanzamiento "mnsubmit" requiere de un *script* con distintas directivas referentes a los recursos, la aplicación, el directorio de trabajo, etc. Estas directivas siguen la sintaxis: $\#@\ directiva = valor$.

Un ejemplo de este *script* se representa en el código 2.2, en el que se observan diferentes partes:

- La directiva "#!/bin/bash", ubicada en la primera linea, que hace alusión al interprete de comandos que se va a utilizar en la ejecución del *script*.
- Seis directivas, situadas desde las lineas 2 a 7, referentes al gestor de colas: "job_name", nombre del trabajo a ejecutar; "initialdir", directorio de ejecución; "output", directorio de almacenamiento de la salida estándar generada por la aplicación; "error", directorio de almacenamiento de la salida de error generada por la aplicación; "total_tasks", número de procesos a ejecutar; "wall_clock_limit", tiempo limite de ejecución.
- En la última linea se ubica el comando "srun", el cual es un comando propio del gestor de colas SLURM usado para la ejecución de aplicaciones paralelas.

```
1 #!/bin/bash
2 #@ job_name = MPI_test
3 #@ initialdir = .
4 #@ output = MPI_%j.out
5 #@ error = MPI_%j.err
6 #@ total_tasks = 32
7 #@ wall_clock_limit = 02:00:00
8
9 srun ./MPI_bin_executable
```

Código 2.2: Ejemplo de *script* de lanzamiento de tareas en Altamira

Configurando correctamente este *script*, el sistema interpretará cada una de las directivas y ejecutará la aplicación acorde a ellas.

CAPÍTULO 3

DISEÑO DEL SISTEMA

En este capítulo se estudiará la arquitectura general, y el proceso de diseño. Para ello se entrará en detalle en los requisitos funcionales y no funcionales, los casos de uso y el problema de codiseño entre el servidor web y el *cluster*.

A lo largo de este y los capítulos restantes se hará un uso frecuente del término "experimento". El uso de este término en este documento esta asociado al proceso de envío de un código fuente junto con la información necesaria a un *cluster*, la compilación de este código en el *cluster*, y su ejecución haciendo uso de Extrae para la captura de las trazas.

3.1 Arquitectura general

El sistema constará fundamentalmente de tres partes, como se representa en la figura 3.1. La primera formada por los equipos con los que los usuarios accederán al servicio. La segunda compuesta por el servidor web. Este equipo será el intermediario entre los usuarios y el *cluster* sobre el que se lanzarán los experimentos. Por último los *clusters*, la parte fundamental de este proyecto, a los que el servidor web se conectará haciendo uso de los protocolos *Secure SHell* (SSH), para acceder de forma remota, y *Secure Transfer File Protocol* (SFTP), para transferir el código fuente de los usuarios y recoger las trazas capturadas.

Para realizar estas acciones el servidor ha de disponer de algún medio de acceso al *cluster*, ya sea porque el *cluster* dispone de una dirección pública o bien que exista una conexión privada entre el *cluster* y el servidor.

3.2 Requisitos del sistema

La idea general del diseño consistía en crear una plataforma capaz de proporcionarles a los usuarios un procedimiento para obtener trazas en sistemas HPC, y un sistema de compartición de trazas a modo de repositorio. A partir de ahí, el primer paso era concretar una serie de requisitos, tanto funcionales como no funcionales, para guiar el diseño. La lista de requisitos que se expone a continuación refleja esta etapa de desarrollo para alcanzar los objetivos propuestos.

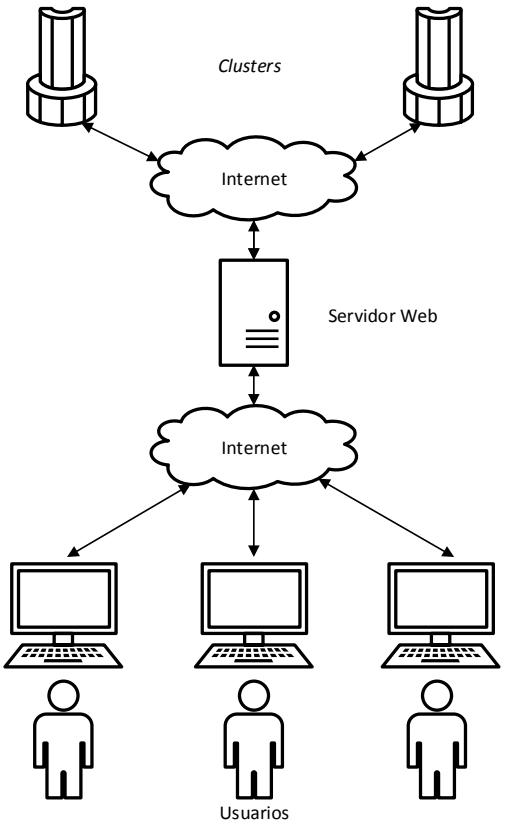


Figura 3.1: Estructura general del sistema, formada por los usuarios, el servidor web y los *cluster*.

3.2.1 Requisitos funcionales

Los requisitos funcionales son aquellos servicios que la plataforma ha de ofrecer para considerar que el sistema cumple con los objetivos marcados. A continuación se describen los requisitos especificados en este sistema.

RF1 - Autenticación

Con motivo de evitar que usuarios malintencionados tengan un punto de acceso a sistemas privados es necesario que la plataforma disponga de un método de control.

Para ello, el sistema requerirá de autenticación a los usuarios para tener pleno acceso a sus funcionalidades, en concreto la realización de experimentos. Los usuarios no autenticados unicamente podrán acceder al listado de trazas.

Si un usuario no esta registrado en el sistema deberá acceder al servicio de registro de usuarios si desea disponer de todas las funcionalidades.

RF2 - Registro de usuarios

Para que exista un subsistema de autenticación es necesario que exista un método por el cual

los usuarios puedan registrase en la plataforma. Este proceso no puede ser totalmente automático dado que se quiere evitar por todos los medios que la plataforma sea empleada para fines inapropiados.

RF3 - Eliminación de usuarios existentes

Dado que un usuario puede tener asociados a su cuenta una serie de aplicaciones y experimentos almacenados en el servidor web, es importante decidir que ocurrirá con estos elementos en caso de que un usuario desee darse de baja de la plataforma. Como uno de los objetivos de este proyecto es crear un repositorio de trazas público es lógico pensar que la mejor opción es mantener las trazas generadas aunque el usuario autor de ellas sea eliminado de la plataforma. En el caso del código fuente de las aplicaciones no es así, ya que se pretende dar al propietario la opción de no permitir a otros usuarios usar un determinado código fuente para que realicen experimentos con él.

RF4 - Gestión de usuarios

En la plataforma coexistirán tres tipos de usuarios, los anónimos, los registrados y los administradores. Es importante dictaminar que acciones podrán llevar a cabo cada uno de estos usuarios. En la tabla 3.1 se recogen los permisos de cada usuario que ha de implementar el sistema.

Funcionalidad	**Usuario anónimo**	**Usuario registrado**	**Administrador**
Solicitud de registro	$\surd$	X	X
Dar de alta a un usuario	X	X	$\surd$
Dar de baja a un usuario	X	X	$\surd$
Subir código fuente	X	$\surd$	$\surd$
Editar código fuente	X	X	$\surd$
Descargar código fuente	$\surd$	$\surd$	$\surd$
Eliminar código fuente	X	$\surd$	$\surd$
Extracción de trazas	X	$\surd$	$\surd$
Añadir trazas	X	$\surd$	$\surd$
Editar trazas	X	X	$\surd$
Eliminar trazas	X	$\surd$	$\surd$
Descargar trazas	$\surd$	$\surd$	$\surd$

Tabla 3.1: Permisos de usuario

RF5 - Subida de código fuente

Para poder extraer la traza de la ejecución de una aplicación es necesario enviar dicha aplicación al sistema HPC. Para ello se transferirá desde el servidor web el código fuente de la aplicación para ser compilado en estos sistemas. De otra forma, si se enviase el código binario ejecutable sería necesario asegurarse de que este es compatible con el sistema en el que se pretende ejecutar. Por ello los usuarios podrán subir los códigos fuentes que deseen al servidor web para su posterior captura de trazas. Además de los ficheros correspondientes al código fuente, se solicitará información que permita caracterizar e indicar a otros usuarios su utilidad y modo de empleo.

Además como se ha adelantado anteriormente existe la necesidad de permitir a los propietarios de un código proporcionarles la opción de compartir con el resto de usuarios sus códigos fuentes. Para ello se definirá un estado de publicación del código fuente. Al subir el código fuente el usuario podrá optar por publicar o no dicho código. En caso de que elija publicarlo el resto de usuarios podrán utilizarlos para capturar sus trazas. En caso negativo, el propietario será el único con acceso a dicho código. Esta opción no podrá ser revertida por el usuario, siendo el único capaz de cambiar el estado de publicación el administrador. En caso de error en la elección el usuario puede ponerse en contacto con el administrador, o bien, eliminar el código fuente y volver a subirlo. Con esto se desea evitar que los usuarios que ya hallan realizado experimentos empleando código fuente de otros usuarios no se encuentren conflictos al intentar acceder a ese código fuente.

RF6 - Eliminación de código fuente

Los códigos fuentes estarán asociados al usuario que los registró en el servidor web. Los códigos fuente almacenados en el servidor unicamente podrán ser eliminados por estos usuarios, a excepción del administrador. Además sólo aquellos códigos fuente que no tengan relación con ningún experimento se podrán eliminar. Con ello se pretende que los usuarios ajenos a un código que lo hayan empleado para generar trazas pierdan este tipo de información y que no se vea afectado su trabajo si el usuario propietario decide eliminar el código.

RF7 - Listado de códigos fuente

Con motivo de poder realizar experimentos con un determinado código fuente, el sistema ha de ser capaz de listar los códigos fuente que puede utilizar, y actualizarlos dinámicamente. Estos códigos serán los que haya registrado en el servidor web el propio usuario y aquellos que hayan sido publicados por otros usuarios.

RF8 - Envío de programas paralelos en el *cluster*

La plataforma ha de ser capaz de enviar el código fuente seleccionado por el usuario, además de información correspondiente a la configuración de la ejecución y de Extrae, para de esta forma poder ejecutar y compilar el código fuente por los motivos anteriormente comentados en el requisito funcional RF5.

Dado que el objetivo final es que el sistema permita la extracción de trazas en distintos *cluster*, además de por su simplicidad, se ha optado por compilar para cada experimento el código fuente, independientemente de que un programa haya sido usado en un mismo *cluster* en repetidas ocasiones.

Los usuarios registrados podrán seleccionar un código fuente para su ejecución y capturar la trazas en el *cluster* que seleccione. Para este lanzamiento el usuario deberá proporcionar información adicional sobre la ejecución y la configuración de Extrae.

RF9 - Selección de una licencia asociada a la utilización de las trazas

Como las trazas que los usuarios creen utilizando la herramienta objeto de este proyecto serán publicadas en un repositorio, es recomendable ofrecer algún método que permita a los autores decidir que tipo de derechos desean ofrecer al resto de usuarios para hacer uso de esta información, bajo ciertas condiciones. Es aquí donde las licencias entran en juego. El sistema

proporcionará al usuario una lista de licencias con distintas condiciones de uso para que sea éste el que elija cómo se han de usar las trazas en caso de que se desee explotar este tipo de información.

RF10 - Compilación y ejecución del código fuente

Con los ficheros de configuración de la ejecución y de Extrae, y del código fuente en el *cluster*, el sistema ordenará la compilación para generar el ejecutable correspondiente a la aplicación y posteriormente lanzará la ejecución, habiéndola configurado para poder hacer uso de Extrae, al gestor de colas.

RF11 - Transferencia de la traza

Los *cluster* presentan cuotas que limitan el espacio disponible de sus usuarios, por esta razón no es razonable almacenar las trazas en estos sistemas, ya que se pueden tratar de archivos con un tamaño considerable.

Por ello, el sistema de forma automática monitorizará el estado de la ejecución de la aplicación lanzada en el *cluster*. En caso de que haya sido completada, se solicitará la transferencia de los ficheros de la traza del *cluster* al servidor web y la registrará en la base de datos junto con información adicional relevante a la ejecución, además del correspondiente fichero de licencia e información sobre la herramienta diseñada en este proyecto.

RF12 - Adición de trazas

Los usuarios podrán enviar trazas al servidor web para que aparezcan en el repositorio de trazas.

Dado que esta función puede usarse para presentar contenido sin relación alguna con las trazas, es necesaria la implementación de algún método que permita filtrar estos contenidos y detectar a estos usuarios. Por ello las trazas solo las publicará en el repositorio el administrador. En caso de que un usuario haya contribuido de forma positiva añadiendo trazas con la posterior aprobación del administrador, el administrador ha de poder indicar al sistema que las siguientes trazas que suba este usuario sean publicadas en el repositorio.

RF13 - Listado de trazas

Para que los usuarios tengan acceso al repositorio de trazas de la plataforma, se listarán en una de las páginas del sitio web.

Se listarán todos los experimentos lanzados, tanto aquellos que aún no se han completado, como los que ya tiene sus correspondientes ficheros de la traza registrados en el servidor además de las trazas enviadas por los usuarios que hayan sido aprobadas por el administrador. Cuando un nuevo experimento sea lanzado o cuando la traza de un experimento sea registrada en el sistema se ha de actualizar esta lista.

RF14 - Adición de citas a las traza

Las citas tienen una gran importancia en el mundo académico y científico, dado que en gran medida es una forma de reconocer el esfuerzo de personas ajenas que han contribuido en la

realización de un trabajo gracias a la publicación de sus conocimientos.

Dentro del listado de trazas se le proporcionará al autor del experimento la opción de añadir una cita sobre el artículo que caracteriza la aplicación que se ha ejecutado a través de la herramienta diseñada. Además se propone que los usuarios, aunque no sea de forma obligatoria, puedan utilizar el formato de la herramienta BibTeX [1] ampliamente utilizado para dar formato a las listas de referencias en la creación de artículos.

3.2.2 Requisitos no funcionales

En los requisitos no funcionales se incluyen aquellas capacidades tanto *hardware* como dependencias *software* que han de tener las distintas partes del sistema para que el funcionamiento de esta plataforma sea posible. A continuación se clasifican según la parte de la arquitectura afectada.

Cluster

Para poder emplear un *cluster* u otro sistema HPC en esta plataforma será necesaria la aprobación de su uso por parte del personal regulador del mismo. Se requerirá de una cuenta de usuario en él, que será empleada por el servidor web para acceder a ese *cluster* o sistema. Esto no implica que el usuario final tenga permisos de ejecución en el *cluster*, sino que el servidor web hará uso de esta cuenta de para ejecutar las aplicaciones de sus usuarios. Además la herramienta Extrae ha de estar disponible y funcional, así como las herramientas necesarias para la compilación de aplicaciones paralelas, en el caso de Altamira se han empleado las bibliotecas de OpenMPI [9]. Para poder compilar y ejecutar los comandos del gestor de colas del *cluster* se hará uso de conexiones remotas seguras siguiendo el protocolo SSH. Para transferir el código fuente del servidor web al *cluster* y las trazas del *cluster* al servidor web se utilizará el protocolo de transferencia segura de ficheros SFTP. El *cluster* ha de ser capaz de poder ejecutar *scripts*, independientemente del lenguaje de programación o interprete de comandos que se emplee, que cubran las distintas fases del proceso diseñado, que se estudiarán en el apartado 4.2.

Servidor web

En el servidor web, será necesaria la instalación de una distribución de Drupal. En concreto la versión 7 es la usada en el desarrollo de este proyecto, y es con la se asegura la compatibilidad del sistema. En [36] y [40] se detalla el proceso de instalación para distintos sistemas operativos. Además, en caso de querer abrir al mundo la plataforma, será necesaria la disponibilidad de un dominio web público, con el que los usuarios de todo el mundo puedan acceder a la plataforma.

Clientes de la plataforma web

Los dispositivos con los que los usuarios accederán al servicio deberán contar con un navegador web que soporte CSS y JavaScript, requisitos de Drupal necesarios para ofrecer todas las funcionalidades del sitio web.

3.3 Casos de uso

En este apartado se describen los casos de uso más relevantes del sistema diseñado. Estos casos de uso indican las posibles interacciones de los usuarios o autores con el sistema. En la tabla 3.2 se describen los distintos tipos de autores que se encontrarán en el sistema.

Actor	**Descripción**
Usuario anónimo	Usuario de la aplicación no registrado en el sistema. Este tipo de usuario unicamente tendrá acceso a la colección de trazas.
Usuario registrado	Usuario de la aplicación registrado en el sistema. Este tipo de usuario podrá añadir código fuente, ejecutar experimentos para la extracción de trazas y añadir trazas, además de visualizar la colección de trazas.
Administrador de contenido	Usuario con permisos de edición, creación, y eliminación de cualquier contenido

Tabla 3.2: Autores de los casos de uso del sistema.

Los casos de uso definidos son: inicio/menú; registro y autenticación de usuarios; repositorio de trazas; añadir código fuente; realizar experimento; añadir trazas. Estos casos de uso se describen a continuación.

3.3.1 Inicio/menú

Con este caso de uso se desea representar las distintas elecciones que podrá realizar un usuario usando los menús de la aplicación. Un usuario anónimo podrá acceder al subsistema de registro y al repositorio de trazas, mientras que uno registrado podrá añadir código fuente, realizar experimentos, añadir trazas, acceder al repositorio de trazas y acceder a la configuración de su cuenta, como se refleja en la figura 3.2.

El flujo de eventos, las distintas fases por las que el actor y el sistema pasarán, en este caso de uso se detalla en la tabla 3.3.

3.3.2 Registro y autenticación de usuarios

En la figura 3.3 se representa el caso de uso “Registro/Autenticación de usuario”. Este caso de uso hace referencia al proceso de registro de un usuario en la plataforma web así como el proceso de autenticación. Un usuario anónimo que quiera hacer uso de las herramientas que la plataforma ofrece para la captura de trazas deberá estar autentificado en el sistema. En caso de que no lo esté, deberá autentificarse, y además si no esta registrado deberá enviar una solicitud de registro. Esta solicitud podrá ser aceptada o denegada por el administrador de contenido. En la tabla 3.4 se recoge la secuencia de eventos dada en este caso de uso.

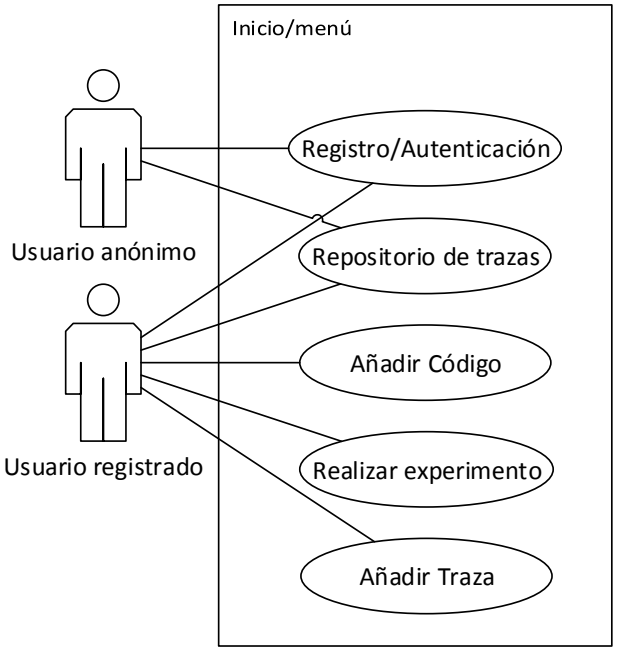


Figura 3.2: Caso de uso del menú de acciones. (Añadir también configuración de la cuenta.)

Actor	**Acción del actor**	**Respuesta del sistema**
Usuario Anónimo	1 - Solicita el acceso al directorio principal del sitio web	2 - Muestra el contenido de la página principal junto con el menú de autenticación, y el menú de navegación con dos opciones, el enlace a la página principal y el enlace al repositorio de trazas.
Usuario Registrado	1 - Solicita el acceso al directorio principal del sitio web	2 - Muestra el contenido de la página principal junto con el menú de configuración de la cuenta de usuario, y el menú de navegación con cuatro enlaces: a la página principal, al formulario de subida de código fuente, al formulario de realización de experimentos, al formulario de subida de trazas y al repositorio de trazas.

Tabla 3.3: Flujo de eventos del caso de uso "Inicio/menú".

3.3.3 Repositorio de trazas

En este caso de uso, figura 3.4, se representa el acceso de cualquier usuario, no importa si esta registrado o no en el sistema, a la colección de trazas generadas con la herramienta diseñada. Además también se desea indicar que el único capaz de eliminar trazas será el administrador de contenido. En la tabla 3.5 se describe el flujo de eventos de este caso de uso.

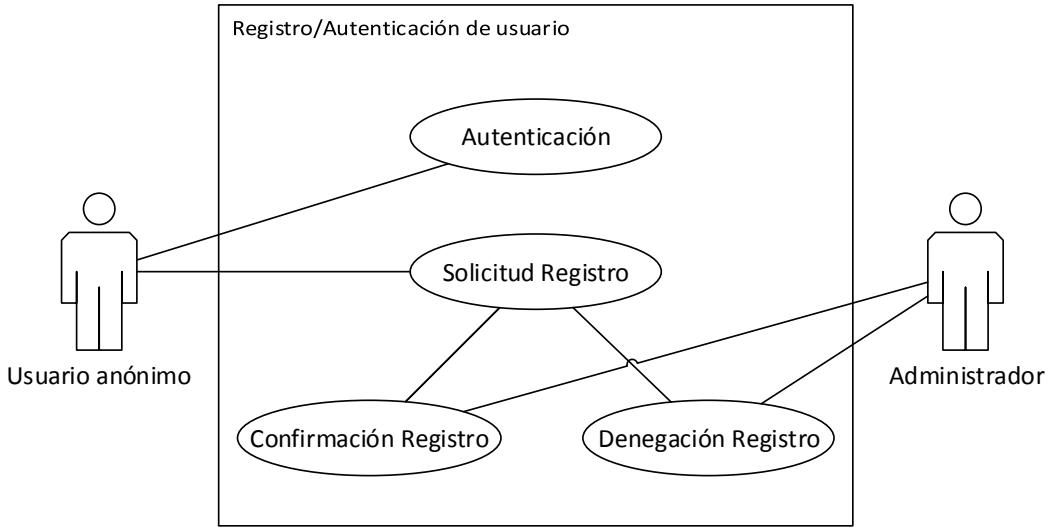


Figura 3.3: Caso de uso del registro o autenticación de un usuario anónimo

Actor	**Acción del actor**	**Respuesta del sistema**
Usuario Anónimo	1 - Introduce nombre de usuario y contraseña en el menú de autenticación, y pulsa el botón de autenticación.	2.1 - En caso de que la combinación de datos sea correcta el usuario inicia sesión, pasando a ser un usuario registrado. 2.2 En caso de que la combinación sea incorrecta se le informa del error mediante un mensaje en pantalla.
Usuario Anónimo	1 - Pulsa sobre la opción de registro en el menú de autenticación. 3 - Rellena los campos y pulsa el botón de envió.	2 - Redirecciona al usuario al formulario de creación de una cuenta de acceso. 4.1 - Comprueba los campos, si son validos se crea una cuenta en estado de bloqueo, con la que no tendrá los permisos de usuario registrado. 4.2 - En caso de que los campos sean inválidos se informa al usuario de los errores.
Administrador	5.1.1 - Comprueba los datos enviados por el usuario, confirma el registro. 5.1.2 Comprueba los datos enviados por el usuario, denega el registro.	6.1.1 - Cambia el estado de la cuenta de bloqueado a activo, teniendo ahora el usuario permisos de usuario registrado. 6.1.2 - La cuenta es eliminada del sistema.

Tabla 3.4: Flujo de eventos del caso de uso "Autenticación".

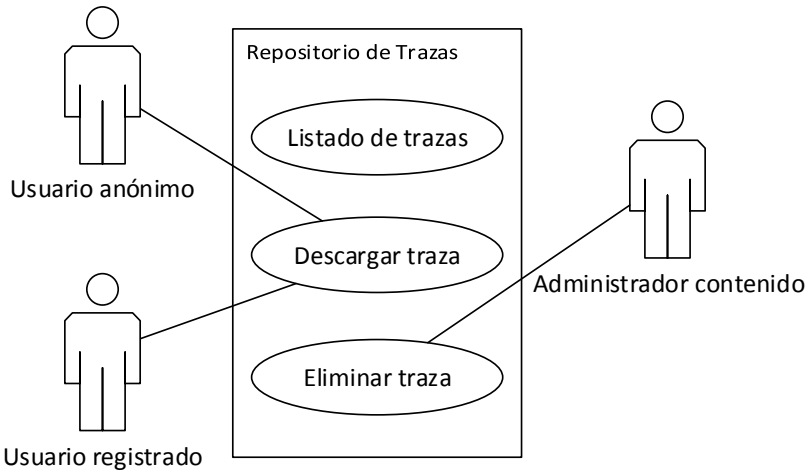


Figura 3.4: Caso de uso del repositorio de trazas

Actor	**Acción del actor**	**Respuesta del sistema**
Cualquier usuario	1 - Envía solicitud de acceso a la dirección del repositorio de trazas 3.1 - Selecciona descargar traza de un experimento. 3.2 - Selecciona un experimento.	2 - Genera una lista de los experimentos realizados. 4.1 - Envía el archivo correspondiente al experimento. 4.2 Redirecciona al nodo asociado al experimento mostrando el valor de sus campos.
Administrador	5.2 - Pulsa el botón de elimación en la página del experimento	6.2 - Elimina el fichero asociado a la traza, sí existe, y elimina el contenido relacionado con el experimento de la base de datos.

Tabla 3.5: Flujo de eventos del caso de uso "Repositorio de Trazas".

3.3.4 Añadir código fuente

Este caso de uso se corresponde con la subida del código fuente de la aplicación a la plataforma web. Al usuario se le mostrará un formulario en el que deberá rellenar cuatro campos: el nombre o título del código fuente, una descripción del programa y su modo de uso, el fichero con el código fuente archivado en un fichero ZIP, y un campo para decidir si el código fuente será publicado para que otros usuarios puedan hacer uso de el o no. En la figura 3.5 se representa este caso de uso y en la tabla 3.6 su flujo de eventos.

3.3.5 Realizar experimento

En este caso de uso el usuario accederá a una lista de códigos fuente disponibles. Esta lista variará según el usuario, ya que a cada uno tendrá disponibles los que él haya añadido y los que otros usuarios hayan publicado. Una vez seleccionado el código fuente se le mostrará al usuario un formulario con distintos campos relacionados con la ejecución de la aplicación y captura de

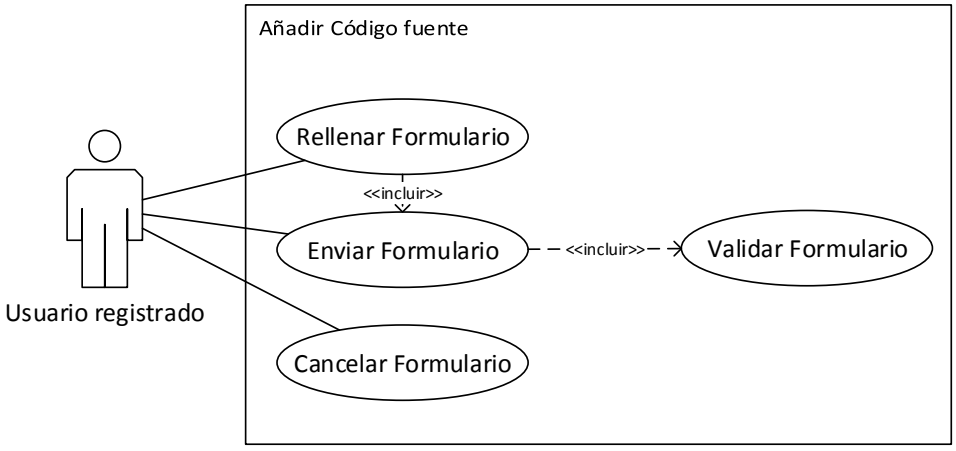


Figura 3.5: Caso de uso para la adición de código fuente

Actor	**Acción del actor**	**Respuesta del sistema**
Usuario registrado	1 - Accede a la dirección de adición de un código fuente.	2 -Redirige a la página web del formulario de creación de un código fuente.
	3.1 - Rellena los campos y envía el formulario.	4 - Comprueba el valor de los campos. 5.1.1 - En caso de que sean validos guarda el contenido de ellos en la base de datos. 5.1.2 - En caso contrario se envía un mensaje indicando el error y se redirige al formulario. 6.1.1 - Si el usuario marco la opción de publicar el código fuente, cambia el estado del nodo a público.
	3.2 - Cancela el formulario.	4.2 - Redirecciona a la página de inicio.

Tabla 3.6: Flujo de eventos del caso de uso "Añadir Código Fuente".

las trazas, así como información general, como el nombre del experimento o el tipo de licencia a usar. La representación del caso de uso se ilustra en la figura 3.6, y su flujo de eventos en la tabla 3.7.

3.3.6 Añadir traza

Este caso de uso se corresponde con la subida de una traza por parte del usuario a la plataforma web. Al usuario se le mostrará un formulario en el que deberá rellenar tres campos: el nombre de la traza, una descripción de la traza y el fichero ZIP con la traza e información adicional si conviene como puede ser el código fuente de la aplicación o una licencia de uso. Después

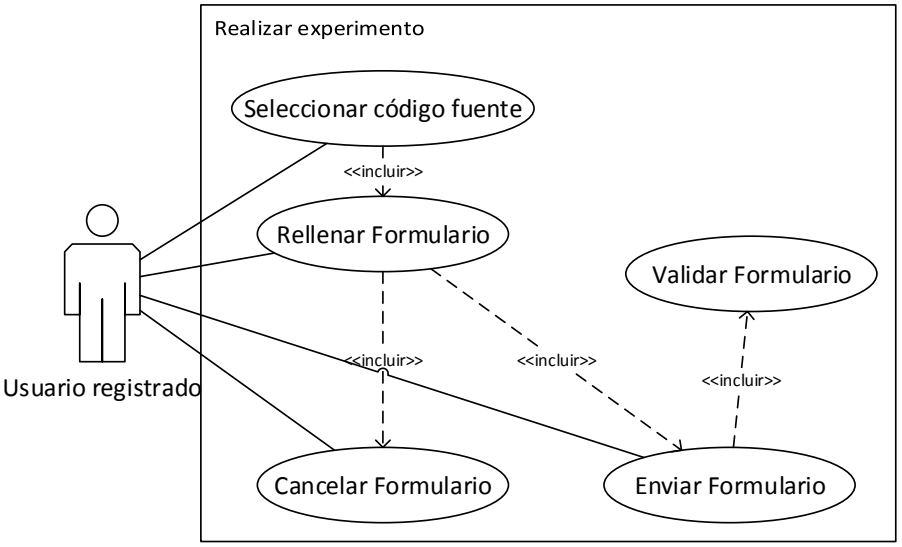


Figura 3.6: Caso de uso para la realización de un experimento

Actor	**Acción del actor**	**Respuesta del sistema**
Usuario registrado	1 - Envía la solicitud de acceso a la dirección de realización de experimentos	2 - Genera una lista de los códigos fuentes disponibles para el usuario. Mostrará aquellos que haya creado el propio usuario y los publicados por otros usuarios.
	3 - Selecciona un código fuente.	4 - Redirecciona a la página del formulario de creación de un experimento.
	5.1 - Rellena los campos y envía el formulario.	6.1.1 - El sistema comprueba los campos, en caso de que sean validos, envía el experimento al *cluster* y almacena en la base de datos la información enviada en el formulario. 6.1.2 - En caso de que no sean validos, redirige al formulario de creación de un experimento indicando los errores.
	3.2 - Cancela el formulario	4.2 - Redirige a la página de inicio.

Tabla 3.7: Flujo de eventos del caso de uso "Realizar Experimento".

será el administrador el que decida si la traza es apta para ser publicada o bien en caso de que el usuario tenga la aprobación del administrador la traza será publicada automáticamente. En la figura 3.7 se representa este caso de uso y en la tabla 3.8 su flujo de eventos.

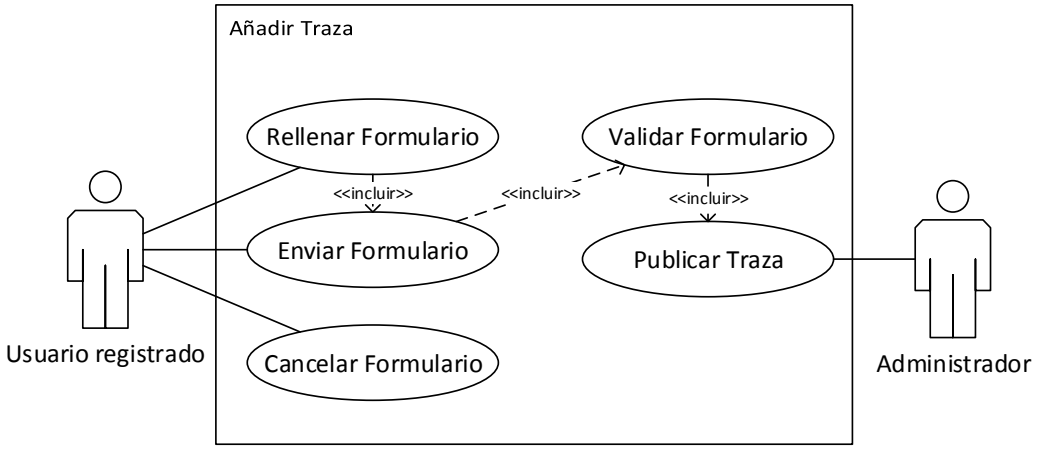


Figura 3.7: Caso de uso para la adición de trazas

Actor	**Acción del actor**	**Respuesta del sistema**
Usuario registrado	1 - Accede a la dirección de adición de una traza. 3.1 - Rellena los campos y envía el formulario. 3.2 - Cancela el formulario.	2 - Redirige a la página web del formulario de creación de trazas. 4 - Comprueba el valor de los campos. 5.1.1 - En caso de que sean validos guarda el contenido de ellos en la base de datos. 5.1.2 - En caso contrario se envía un mensaje indicando el error y se redirige al formulario. 6.1.1 - Si el usuario tiene la autorización del administrador para que sus trazas se publiquen, cambia el estado del nodo correspondiente a la traza a público. 4.2 - Redirecciona a la página de inicio.
Administrador	6.1.2 - Si la traza no es pública y considera que es adecuada para su publicación, envía la solicitud de publicación de la traza.	7.1.2 - Cambia el estado de publicación del nodo de la traza a público.

Tabla 3.8: Flujo de eventos del caso de uso "Añadir Traza".

3.4 Codiseño servidor web-*cluster*

Un aspecto de gran relevancia en el proceso de desarrollo de este proyecto en el que dos sistemas han de interaccionar entre sí para lograr las exigencias de la aplicación, es delimitar las partes funcionales de cada uno de ellos, como se representa en la figura 3.8.

La elección tomada ha sido separar la funcionalidad en dos grandes partes acorde a los dos

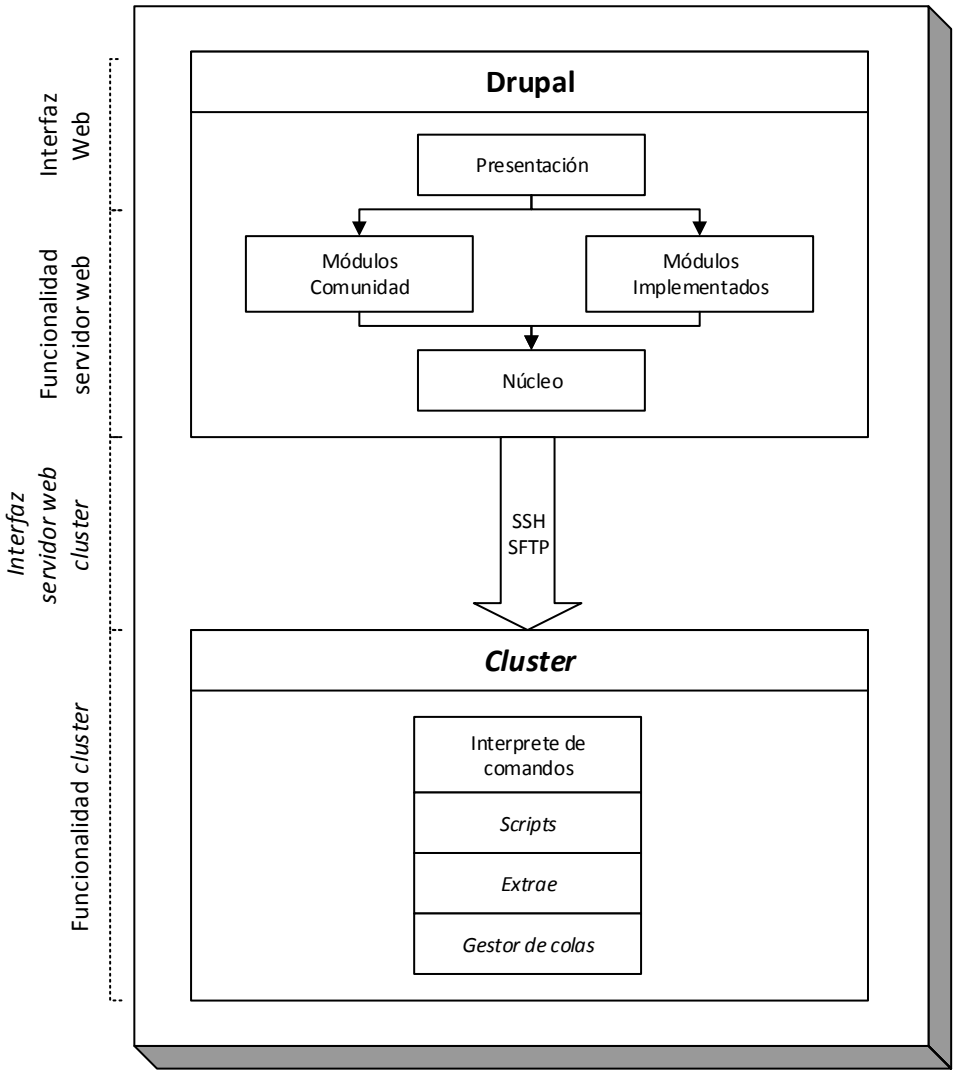


Figura 3.8: Estructura del diseño.

sistemas presentes en la plataforma. Por el lado de los *cluster* se observa que cada uno de ellos requerirá de una implementación conforme a su gestor de colas, instalación de extrae, sistema operativo, etc. Por el otro lado, la parte de Drupal, la gestión de usuarios y de contenido, es similar independientemente del *cluster*.

De esta forma en Drupal, se implementará todo lo relevante al uso de la aplicación por parte de los usuarios, es decir implementará todos los requisitos funcionales salvo el RF10, y en los *cluster* se implementará la funcionalidad relevante a la compilación y ejecución de la aplicación, y la extracción de las trazas de forma independiente para cada uno de ellos, correspondiente a los requisitos funcionales RF10 y RF11. Para que estos sistemas puedan comunicarse entre sí, en los *cluster* se creará un directorio con una estructura definida, como se verá en el apartado 4.2.1, con una serie de *scripts* que proporcionarán dicha funcionalidad. Desde el servidor web, Drupal invocará a estos *scripts* según las acciones de los usuarios o del propio sistema.

CAPÍTULO 4

IMPLEMENTACIÓN

En esta capítulo se estudiará el proceso de implementación de la aplicación. Este proyecto se ha centrado en la implementación en un solo *cluster*, siendo la funcionalidad principal similar con múltiples *clusters*. Al final de este capítulo se explicarán los cambios a realizar para poder trabajar con más de un *cluster*.

4.1 Implementación de la Funcionalidad en Drupal

A Drupal le corresponde la implementación de la interfaz de usuario de la herramienta, así como el control de la ejecución de las aplicaciones en el *cluster*. Para poder solventar las necesidades de la herramienta se ha hecho uso de varios de los módulos del propio núcleo de Drupal, de varios módulos creados por su comunidad y de la creación de un módulo propio. Además en ocasiones se ha hecho uso de la interfaz web que proporcionan estos módulos en el menú de administración del sitio web creado con Drupal y en otras ocasiones se ha hecho uso de las APIs que proporcionan estos módulos, tanto los del núcleo de Drupal como los de la comunidad, en el módulo creado.

El desarrollo de esta parte, correspondiente al servidor web, se ha realizado sobre una maquina virtual por la flexibilidad que ofrece, pudiéndose migrar con gran facilidad a otros equipos en los que desplegar el servidor web asignándole un nombre e IP pública.

Este apartado se estructurará según los requisitos funcionales que solventará el proceso analizado. Drupal se encargará de cubrir todos los requisitos funcionales menos el RF10 - Compilación y ejecución del código fuente y el *cluster* de los requisitos funcionales RF10 y RF11.

4.1.1 Autenticación, registro, eliminación y gestión de usuarios. (RF1, RF2, RF3, RF4)

Drupal cuenta con un módulo denominado “user”. Este módulo pertenece al núcleo de Drupal y proporciona un sistema de registro, y de inicio y cierre de sesión. También permite configurar roles de usuario para clasificar a los usuarios y asignar permisos a estos roles.

Autenticación (RF1)

En primer lugar este módulo define un bloque que puede ser colocado en las distintas regiones de la página definidas en el tema, como se vio en el apartado 2.1.2, y un menú. El bloque, con nombre "User login" se corresponde con los campos, enlaces y botones que se muestran en la figura 4.1. A través de este bloque los usuarios podrán introducir sus combinaciones de nombre de usuario y contraseña para iniciar sesión, o bien acceder a las páginas de registro o de solicitud de una nueva contraseña. El menú, que recibe el nombre de "User menu" contendrá dos enlaces. El primer enlace "my account" referencia a la página de edición de la cuenta de usuario, y el segundo a la página de "logout", en la que se procesará el cierre de la sesión del usuario.

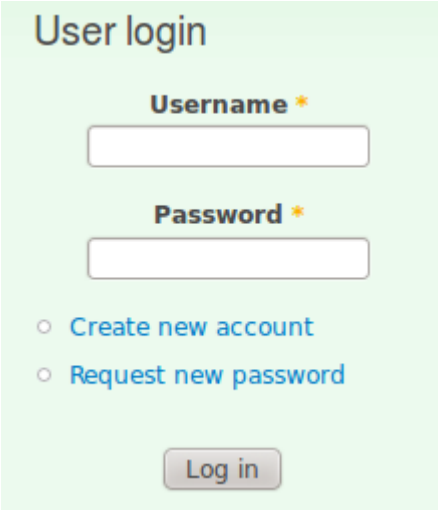


Figura 4.1: Bloque "User login" en el tema "Garland".

Registro y Eliminación de usuarios (RF2 y RF3)

Según el requisito funcional RF2 se argumentaba la necesidad de un sistema de registro de usuarios no automático. En este extremo cabe la posibilidad de crear un sistema en el que el único capaz de crear cuentas de usuario sea un administrador, pero entonces la intención de crear un sistema público en el que los investigadores puedan compartir sus trazas se vería muy limitado. Por estas razones se ha escogido una opción intermedia, en el que cualquier usuario pueda solicitar la creación de una cuenta de usuario a través del servicio web, presentando la correspondiente información personal así como su acreditación de investigación. Posteriormente esta información será revisada por un administrador y se validará o denegará la activación de la cuenta según convenga.

El requisito funcional RF3, resaltaba la necesidad de dar una solución sobre qué hacer con los contenidos creados por un usuario que desee darse de baja. Dado que algunos códigos fuente pueden estar declarados como privados, se ha optado por deshabilitar la cuenta ante la solicitud su solicitud de baja. De esta forma todo su contenido permanecerá en el sistema y será posteriormente cuando un administrador se pondrá en contacto con el usuario para determinar que acción se llevará a cabo con esos códigos, eliminarlos o hacerlos públicos.

Para implementar estas dos soluciones se ha recurrido al menú de administración que pro-

porciona el módulo “user” y se han seleccionado las opciones correspondientes como se muestra en la figura 4.2.

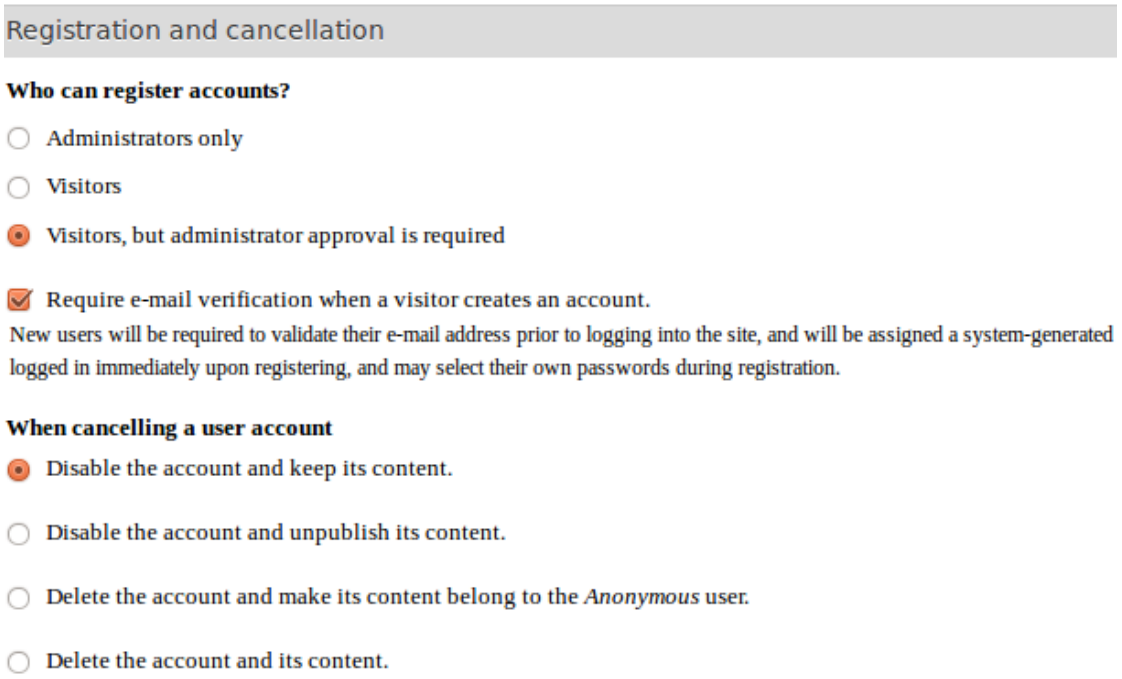


Figura 4.2: Fragmento correspondiente al menú de administración de las cuentas de usuario.

Gestión de usuarios (RF4)

En cuanto a la gestión de usuarios, es necesario definir dos tipos de usuario con diferentes permisos, además del administrador. Por una parte los usuarios anónimos tendrán acceso al repositorio de trazas donde se listarán todas las trazas disponibles, pero no podrán crear, ni modificar contenido de tipo “Source Code”, “Experiment” y “Trace” que como se estudiará estos tipos de contenido se emplearán para almacenar el código fuente, las trazas generadas por la plataforma y las trazas enviadas por los usuarios, en el servidor web. Por otra parte, los usuarios registrados, podrán crear y eliminar los contenidos de tipo “Source Code” y “Trace” que sean de su propiedad, pero no podrán editarlos, al igual que no podrán editar ni eliminar los contenidos de tipo “Experiment” que hayan sido creados por ellos. Estas acciones solo podrán ser llevadas a cabo por el administrador de contenido. En caso de repetición, o fallo en la captura de la traza se deberá solicitar al administrador su eliminación. En la figura 4.3 se muestra el fragmento de la página de definición de roles que permite la edición y creación de roles. En esta imagen se distinguen tres roles: “anonymous user”, “authenticated user” y “administrator”, que se corresponden con usuario anónimo, usuario registrado y administrador del requisito funcional RF4. Posteriormente para definir los permisos de cada usuario basta con acceder a la página de permisos y marcar las opciones de interés. En la figura 4.4 se representa la parte del contenido de esta página correspondiente a los nodos de tipo “Source Code” y “Experiment”.

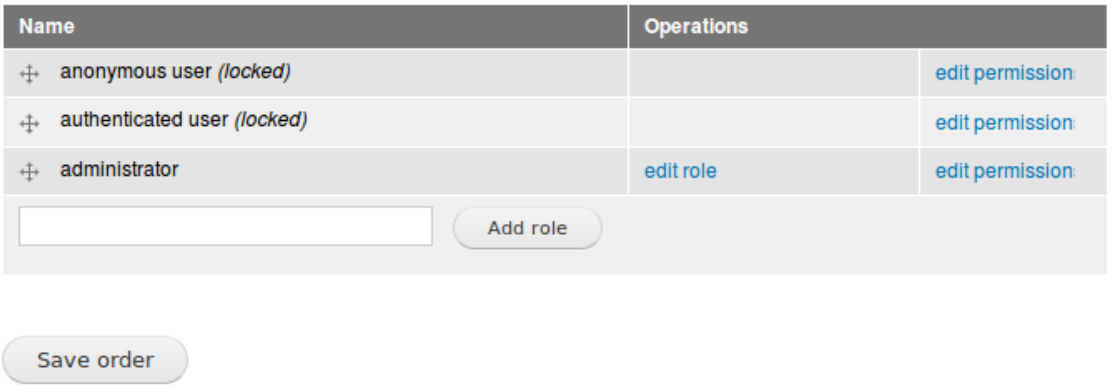


Figura 4.3: Página de definición de roles.

Permission	anonymous user	authenticated user	administrator
Experiment: Create new content	☐	☑	☑
Experiment: Edit own content	☐	☐	☐
Experiment: Edit any content	☐	☐	☐
Experiment: Delete own content	☐	☑	☑
Experiment: Delete any content	☐	☐	☐
Source Code: Create new content	☐	☑	☑
Source Code: Edit own content	☐	☐	☐
Source Code: Edit any content	☐	☐	☐
Source Code: Delete own content	☐	☑	☑
Source Code: Delete any content	☐	☐	☐

Figura 4.4: Página de permisos de usuario.

4.1.2 Subida de código fuente (RF5)

Como ya se ha comentado en varias ocasiones, Drupal permite crear distintos tipos de contenido. Esta funcionalidad esta integrada en el módulo del núcleo “node” que controla la creación, edición, borrado, configuración, y visualización del contenido principal del sitio web. En concreto este módulo gestiona los nodos, que son una clase de objeto que hace referencia a un tipo de contenido, y que típicamente son mostrados como las páginas del sitio web. Este tipo de objetos esta caracterizado por un título, meta-datos (autor, fecha de creación, tipo de contenido) y otros campos opcionales que son controlados por otro módulo del núcleo llamado “fields”, encargado de gestionar los distintos campos o registros del sitio web. Por lo tanto para que los usuarios puedan añadir código fuente, se ha definido un nuevo tipo de nodo o contenido con la información necesaria para identificar un código fuente. En el diseño se le ha dado a este tipo de contenido el nombre de “Source Code” y está compuesto por cuatro campos, además de los meta-datos anteriormente comentados, los cuales se listan a continuación:

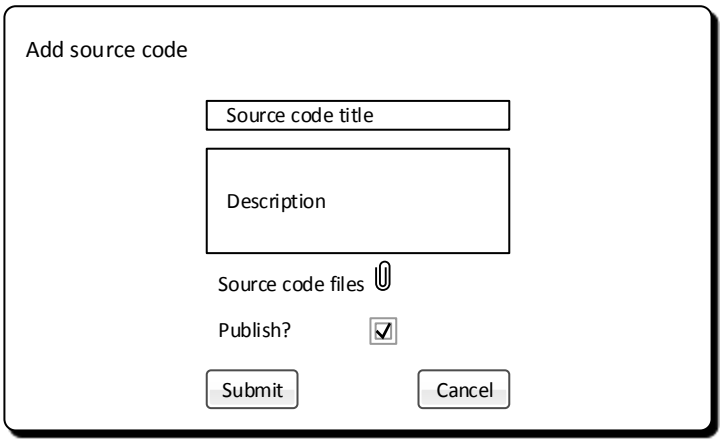


Figura 4.5: Formulario de creación del contenido de tipo “Source Code”.

1. *Title*. Nombre del nodo.
2. *Body*. Descripción del código fuente. En él se recomienda explicar que utilidad tiene el código fuente a añadir, indicaciones de uso, etc, para que otros usuarios puedan hacer uso de este contenido.
3. *Source code*. Ficheros del código fuente. En este campo se añadirá el código fuente comprimido en formato ZIP [35], formato elegido por su extendido uso. Este formato será especificado en la configuración del contenido para que Drupal compruebe la extensión del fichero y en caso de que el fichero enviado no sea de tipo ZIP se cancele el envío del formulario.
4. *Publish*. Si se activa este campo este código fuente será publicado, permitiendo al resto de usuarios su utilización para la realización de experimentos.

Definida esta clase de nodo, los usuarios, si tienen los permisos necesarios, podrán acceder a la página de creación de contenido de este tipo de nodo en las que podrán rellenar estos cuatro campos en un formulario como el representado en la figura 4.5.

Para lograr el propósito del campo “publish” ha sido necesaria la inclusión del proceso reflejado en el diagrama 4.6, en un modulo propio. En este diagrama se destaca el *hook* empleado, “hook_node_presave($node)”, que será invocado por el núcleo de Drupal al enviar el formulario de creación del código fuente. El primer paso del procedimiento consiste en comprobar el tipo de nodo ha crear, en caso de ser “Source Code” se analizará el estado del campo “publish” y se editará la propiedad correspondiente al estado de publicación del nodo según ese valor.

4.1.3 Eliminación de código fuente (RF6)

Para resolver la problemática de este requisito funcional el primer paso que se ha llevado a cabo ha sido configurar los permisos de usuario para que unicamente los propietarios del contenido de tipo “Source Code” sean capaces de eliminar esos nodos. El siguiente paso consiste en comprobar si el nodo a eliminar es o ha sido empleado en la realización de algún

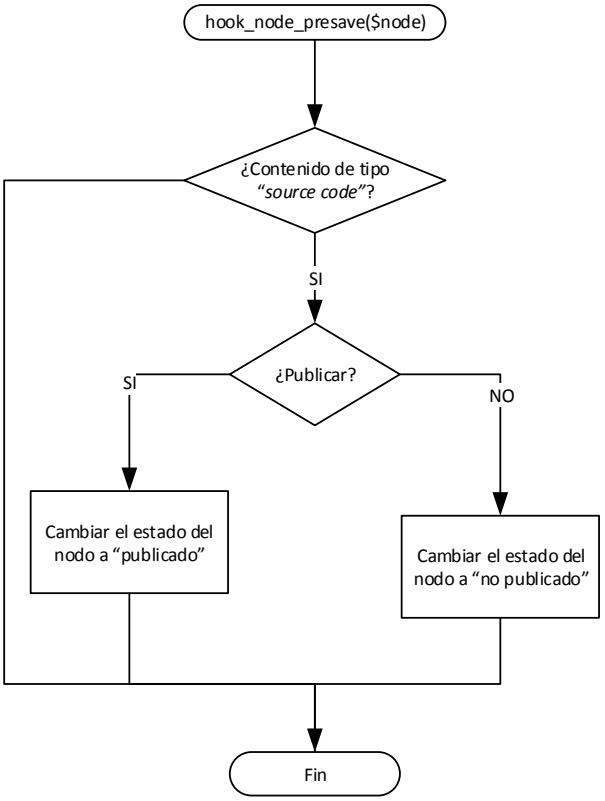


Figura 4.6: Diagrama de flujo para la publicación de un código fuente.

experimento. Para ello se ha añadido el código necesario al módulo creado para implementar la función representada en el diagrama 4.7. En esta implementación se ha utilizado el *hook* "hook_node_delete($node)" que es invocado justo antes de que el nodo sea eliminado de la base de datos cuando el usuario ha seleccionado esta acción. De esta forma, antes de la eliminación del nodo, se puede comprobar si su identificador está asociado a algún contenido de tipo "Experiment". Es decir si alguna traza ha sido generada usando el código fuente que se desea eliminar. En caso afirmativo se denegará la acción y se le enviará un mensaje de error al usuario indicándole que no se puede realizar la acción de eliminación, ya que en ese caso se perdería esta relación de cara al resto de usuarios que hayan podido hacer uso de este código fuente.

4.1.4 Listado de códigos fuente (RF7)

Dado que el sistema ha de mostrar los códigos fuente que puede utilizar un usuario en el sistema, esta lista ha de ser dinámica. Para ello se ha empleado el módulo "views" [41]. Este módulo permite crear listas dinámicas empleando la información almacenada en la base de datos del sitio web. En estas listas se puede seleccionar qué tipo de contenido se ha de incluir, qué criterios han de cumplir los distintos campos del contenido a representar, y el orden en el que se representarán. Además estas listas pueden mostrarse, por ejemplo, como una página web o un bloque, y con distintos formatos, como puede ser en forma de tabla, una lista HTML, o un menú. En la figura 4.8 se muestra la página principal de configuración de la "view" creada para listar los códigos fuente. En ella se puede apreciar que esta "view" es representada como

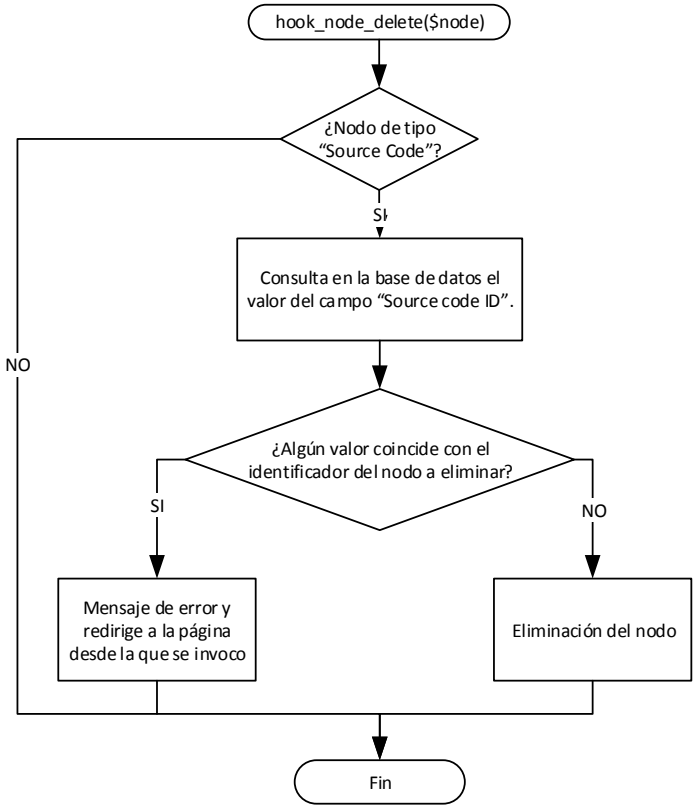


Figura 4.7: Diagrama de flujo del proceso de comprobación de la utilización del código fuente.

una página y en formato tabla. Se mostrarán seis campos que serán el título, la descripción, el fichero con el código fuente y el nombre de usuario del autor del nodo, además de un enlace a la página de eliminación del nodo y un campo denominado "run experiment" el cual se estudiará a continuación.

El campo "run experiment" se trata de un campo personalizado, creado a partir de la API [13] del módulo "views". El propósito de este campo es el de que el usuario pulse sobre este enlace y sea redireccionado a la página de creación de un nodo de tipo "Experiment" asociando a este experimento el código fuente seleccionado. Así cuando el usuario envíe relleno el formulario del experimento el sistema tendrá conocimiento de que código fuente a de enviar al *cluster*. Para hacer esto se ha definido el nuevo tipo de campo indicando que se trata de un enlace URL, que indica la dirección al formulario de creación de contenido de tipo "Experiment". Además el valor del identificador del nodo correspondiente es transferido por medio de la variable $_GET. Esta es una variable predefinida de PHP que permite el intercambio de parámetros entre distintas páginas de un sitio web por medio de la URL.

Por último es necesario filtrar el contenido "Source Code" a mostrar. Como se indicó, un usuario únicamente podrá emplear el código fuente subido por el mismo o aquel código fuente que haya sido publicado. Como se puede apreciar de nuevo en la figura 4.8 en la sección "Filter criteria" se ha definido la siguiente condición:

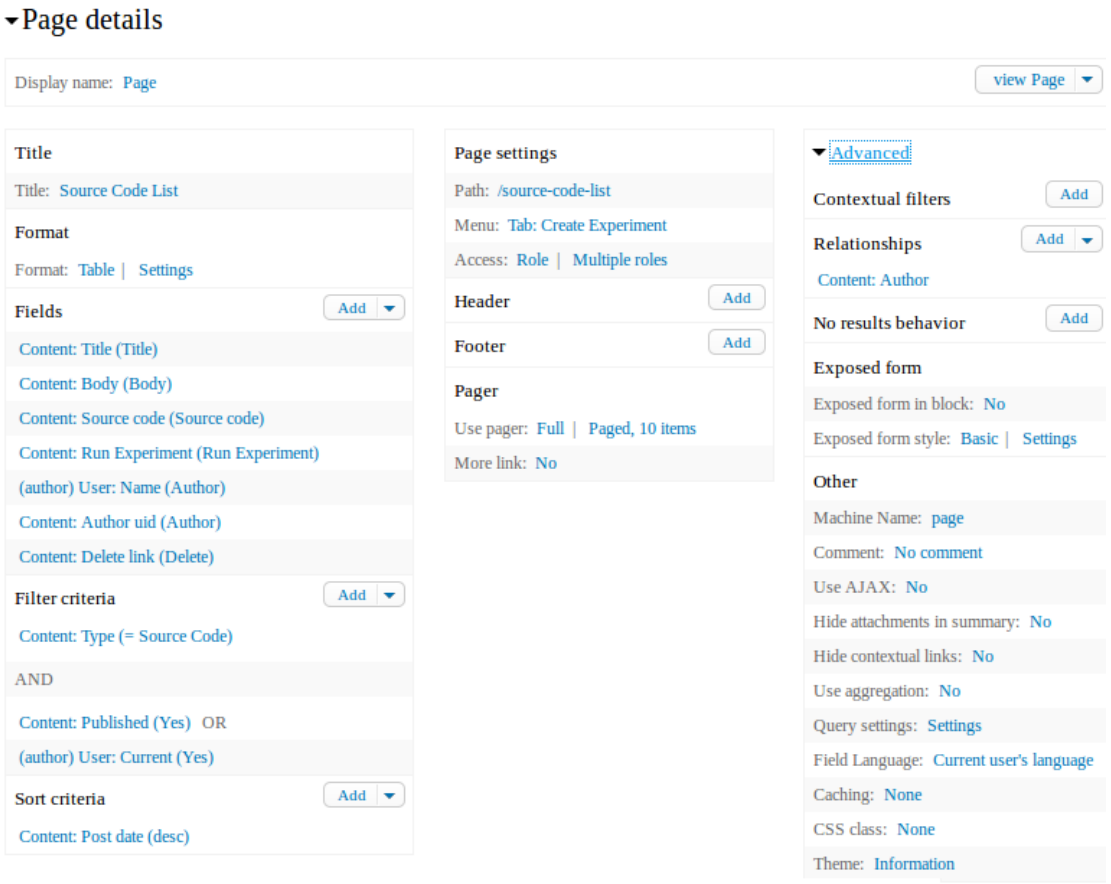


Figura 4.8: Página de configuración principal del "view" "Source Code List"

Content type (= Source Code)
AND
Content: Published (Yes) OR
(author) user : current (Yes)

Esto significa que si el contenido es de tipo “Source Code”, y es publicado o el usuario actual es el usuario autor del contenido se mostrará en el resultado final dicho contenido.

4.1.5 Ejecución de programas paralelos en el *cluster* (RF8)

Al igual que ocurrió con el código fuente, se ha definido un nuevo tipo de contenido denominado “Experiment”. Este tipo de contenido esta caracterizado por varios campos categorizados en tres grupos: opciones de ejecución y caracterización del experimento, opciones de extrae, y argumentos de ejecución. Estos campos permitirán al usuario introducir toda la información que posteriormente será necesaria en el proceso de captura de la traza de la aplicación y caracterizar el experimento. El formulario diseñado para la creación de este tipo de contenido se representa en la figura 4.9.

En el primer grupo, opciones de ejecución y caracterización del experimento (“Experiment Options”), se agrupan los siguientes campos:

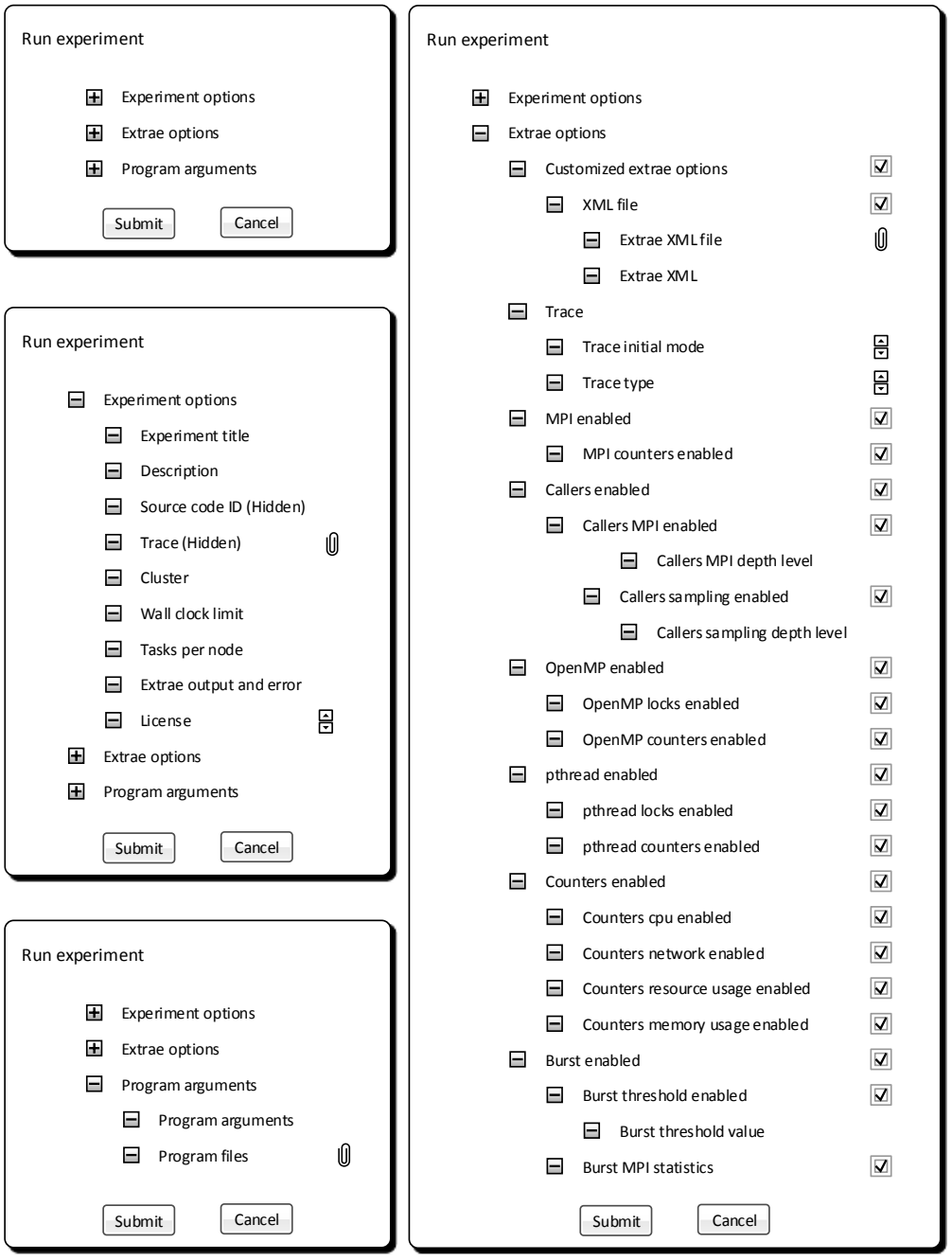


Figura 4.9: Formulario para la creación de un experimeto.

1. *Experiment title*. Título o nombre del experimento.
2. *Description*. Descripción del experimento. Información que el autor quiera añadir sobre el experimento para que otros usuarios puedan entender el propósito del mismo.
3. *Source code ID*. Identificador del nodo correspondientes al código fuente de la aplicación de la que se va a realizar la extracción de trazas.
4. *Trace*. Ficheros correspondientes a la traza de la ejecución e información adicional del experimento, como la licencia o las salidas estándar y de error resultantes de la ejecución.
5. *Cluster*. *Cluster* sobre el que se lanzará el experimento.
6. *Wall clock limit*. Tiempo máximo de ejecución de la aplicación en el *cluster*.
7. *Total tasks*. Número total de procesos de la ejecución.
8. *Tasks per node*. Número de procesos de la ejecución por nodo del *cluster*.
9. *Extrae output and error*. Marcando este campo se guardará las salidas estándar y de error de la ejecución junto con los ficheros de la traza.
10. *License*. Licencia asociada a la traza.

En el segundo grupo se encuentran las opciones de extrae (“Extrae Options”). No todas las opciones de Extrae están presentes. Esto se debe a la gran cantidad de secciones que componen el fichero XML, así como la complejidad de implementar en un formulario algunas de las opciones anidadas con las que cuenta la configuración de Extrae.

La intención es ofrecer al usuario tres métodos diferentes desde los que configurar la extracción de las trazas con las posibilidades que Extrae otorga.

- La primera consiste en permitir al usuario editar un fichero XML, a modo de plantilla, en linea. Es decir que el usuario pueda escribir o modificar un fichero de configuración de Extrae presente en el formulario. Para tomar esta opción el usuario ha de marcar el campo “Customized extrae options” sin marcar el campo “XML file”, que significan que se utilizará una configuración personalizada y que no se subirá el fichero XML como se verá a continuación. De esta forma el campo “Extrae XML” estará habilitado mientras que el resto de opciones no.
- La segunda opción consiste en que el usuario suba al servidor web su propio archivo de configuración. Para ello el usuario ha de marcar los campos “Customized extrae options” y “XML file”, que habilitaran el campo “Extrae XML file” para que el usuario pueda adjuntar su fichero de configuración.
- La tercera opción ofrece al usuario un formulario con el que configurar algunas secciones del fichero XML. Esta opción es la más sencilla de cara al usuario pero a su vez la de mayor limitación ya que no todas las opciones de las que dispone Extrae están presentes.

De esta forma los usuarios con experiencia utilizando Extrae podrán ajustar su configuración a medida y los usuarios más noveles podrán adentrase en las características de la herramienta de una forma comoda y sencilla.

En cuanto al tercer grupo, los argumentos del programa (“Program Arguments”), se agrupan dos campos:

1. *Program argument*. Los argumentos del programa, es decir los argumentos que el usuario incluiría junto con el comando de ejecución del programa, en caso de que la aplicación a ejecutar requiera de ellos.

2. *Program files*. Los ficheros de entrada que la aplicación necesite. En caso de que sea así en el campo "Program argument" se ha de escribir el nombre del o los ficheros subidos al servidor web. Estos ficheros se subirán en formato ZIP y la referencia que se haga a ellos a través del campo "Program argument" tendrá que ser correcta. Para ello habrá que usar la ruta relativa a los ficheros desde el directorio raíz del archivo ZIP.

Para implementar estas ideas se ha recurrido a cuatro módulos desarrollados por la comunidad de Drupal y una libreria PHP.

El primer módulo "Field group" [43] permite agrupar distintos campos pudiendo estructurar el formulario según se representa en la figura 4.9. Es decir permite organizar el formulario en distintas secciones. Este módulo añade elementos a los que Drupal proporciona por defecto en la creación de tipos de contenido. Con estos elementos se pueden agrupar los campos de diferentes formas; con menús desplegables, pestañas horizontales y verticales, etc. En la figura 4.10, se muestra la página de configuración de los campos asociados al contenido de tipo "Experiment". En ella se pueden apreciar dos tipos de elementos, representados en letra negrita se encuentran los grupos, y en letra normal los campos, proporcionados por defecto por Drupal. Como se aprecia en el fragmento hay tres niveles de indentación, que viene a representar la jerarquía de cada elemento. En el nivel más alto se encuentra el grupo "Vertical" en el que se encuentran los tres grupos ya comentados en este subapartado; "Experiment Options", "Extrae Options" y "Program Arguments". Estos grupos serán mostrados en un menú vertical formado por tres pestañas según aparece indicado en la columna "widget" de la figura.

Label	Machine name	Field type	Widget	Operations	
Vertical	group_vertical	Vertical tabs group	**tabs classes** group-vertical field-group-tabs		delete
Experiment Options	group_experiment_options		**tab** closed **required_fields** yes		delete
Experiment name	title				
Description	body		Text area with a summary	edit	delete
Source code	field_source_code		File	edit	delete
Trace	field_trace		File	edit	delete
Cluster	field_cluster	List (text)	Select list	edit	delete
Wall clock limit	field_wall_clock_limit	Text	Text field	edit	delete

HTML element
Div
HTML5
Fieldset
Vertical tabs group
Vertical tab
Horizontal tabs group
Horizontal tab
Multipage group
Multipage
Accordion group
Accordion item

Figura 4.10: Fragmento de la página de configuración de los campos del contenido de tipo "Experiment".

El segundo módulo tiene el nombre de "Conditional fields" [39], y permite mostrar/ocultar, habilitar/deshabilitar, además de otras opciones, campos a partir del valor de otro campo, es decir se podrá cambiar el estado de un campo dependiendo del valor de otro. Este módulo proporciona una interfaz incluida en la definición de tipos de contenido que permite definir las distintas relaciones entre campos, como se representa en la figura 4.11. Por ejemplo, atendiendo a la primera fila se observa que el campo "Trace type" depende del campo "Customized extrae options" estando habilitado cuando este tiene valor '0', es decir, cuando la casilla no este

marcada.

Dependent	Dependees	Description		Operations	
Trace type (field_trace_type)	Customized extrae options (field_extrae_options)	*Trace type* is enabled when *Customized extrae options* has value "0".		edit	delete
Trace initial mode (field_trace_initial_mode)	Customized extrae options (field_extrae_options)	*Trace initial mode* is enabled when *Customized extrae options* has value "0".		edit	delete
Callers MPI depth level (field_callers_mpi_depth_level)	Callers enabled (field_callers_enabled)	AND	*Callers MPI depth level* is enabled when *Callers enabled* has value "1".	edit	delete
	Callers MPI enabled (field_callers_mpi_enabled)		*Callers MPI depth level* is enabled when *Callers MPI enabled* has value "1".	edit	delete
	Customized extrae options (field_extrae_options)		*Callers MPI depth level* is enabled when *Customized extrae options* has value "0".	edit	delete
	Callers enabled (field_callers_enabled)		*Callers sampling depth level* is enabled when *Callers*	edit	delete

Figura 4.11: Fragmento de la página de configuración de las dependencias entre campos del contenido de tipo “Experiment”.

El tercer módulo empleado es “XML Field” [37] que añade un nuevo tipo de campo. Este campo es de tipo texto, que tendrá la peculiaridad de mostrar el código XML con estilo, es decir diferenciando las palabras clave del lenguaje, los comentarios, los valores numéricos, las cadenas de caracteres, etc. Este módulo requiere de una biblioteca PHP con nombre “CodeMirror” [10]. En Drupal es necesaria la instalación del módulo “Libraries API” [38], que añade un nuevo repositorio a Drupal, en el que colocar las bibliotecas adicionales necesarias por los módulos. En la figura 4.12 se muestra un fragmento de la página de configuración de los campos del tipo de contenido “Experiment” en el que aparece el campo “Extrae XML” que hace uso de este módulo para permitir al usuario editar el archivo de configuración de Extrae en linea.

Label	Machine name	Field type	Widget	Operations	
Extrae xml	field_extrae_xml	XML	CodeMirror XML	edit	delete

Figura 4.12: Campo “Extrae xml” en la página de configuración de los campos del contenido de tipo “Experiment”.

Con la parte correspondiente a la interfaz de usuario completada, a continuación se abordará el aspecto funcional. Este contexto se puede dividir en dos partes. La primera está asociada a la implementación de las distintas opciones que puede escoger el usuario en la realización de un experimento: qué licencia, qué método se ha escogido para definir las opciones de Extrae en el fichero XML, etc. La segunda parte se corresponde con el procedimiento de envío de los ficheros del experimento al *cluster*.

En la figura 4.13 se representa el procedimiento para la realización de un experimento en la parte del servidor web que se ha desarrollado por completo en PHP. Partiendo de que un usuario envía la solicitud de realización de un experimento, Drupal invocará a este procedimiento antes de almacenar la información del nodo en la base de datos, dado el uso del *hook* “hook_node_insert($node)”. Este *hook* tiene un parámetro de entrada “node” el cual contiene toda la información asociada al nodo, su identificador, el usuario propietario, los campos asociados a él, el estado de publicación, y otro tipo de información. En el caso de este procedimiento, el tipo de nodo, el valor de los campos rellenados por los usuarios y el identificador de nodo serán suficientes. Lo primero es comprobar si el contenido es de tipo “experiment”, en caso afirmativo se establecerán la conexión SFTP para la transferencia de archivos, y se generarán y

enviarán los dos ficheros de configuración, el de ejecución, y el de Extrae, junto con el código fuente del programa. Después se comprobará si el programa requiere de ficheros de entrada y en caso positivo se enviarán. Por último se establecerá la conexión SSH para la ejecución del *script* "experiment_launcher.py" que requerirá como parámetro de entrada el identificador de nodo del experimento. Hecho esto se cerrará la sesión SSH.

Para el establecimiento de las sesiones SFTP y SSH se ha hecho uso de la biblioteca "php-seclib" [18] y de nuevo del módulo "libraries" que permite cargar bibliotecas ajenas a Drupal desde los módulos.

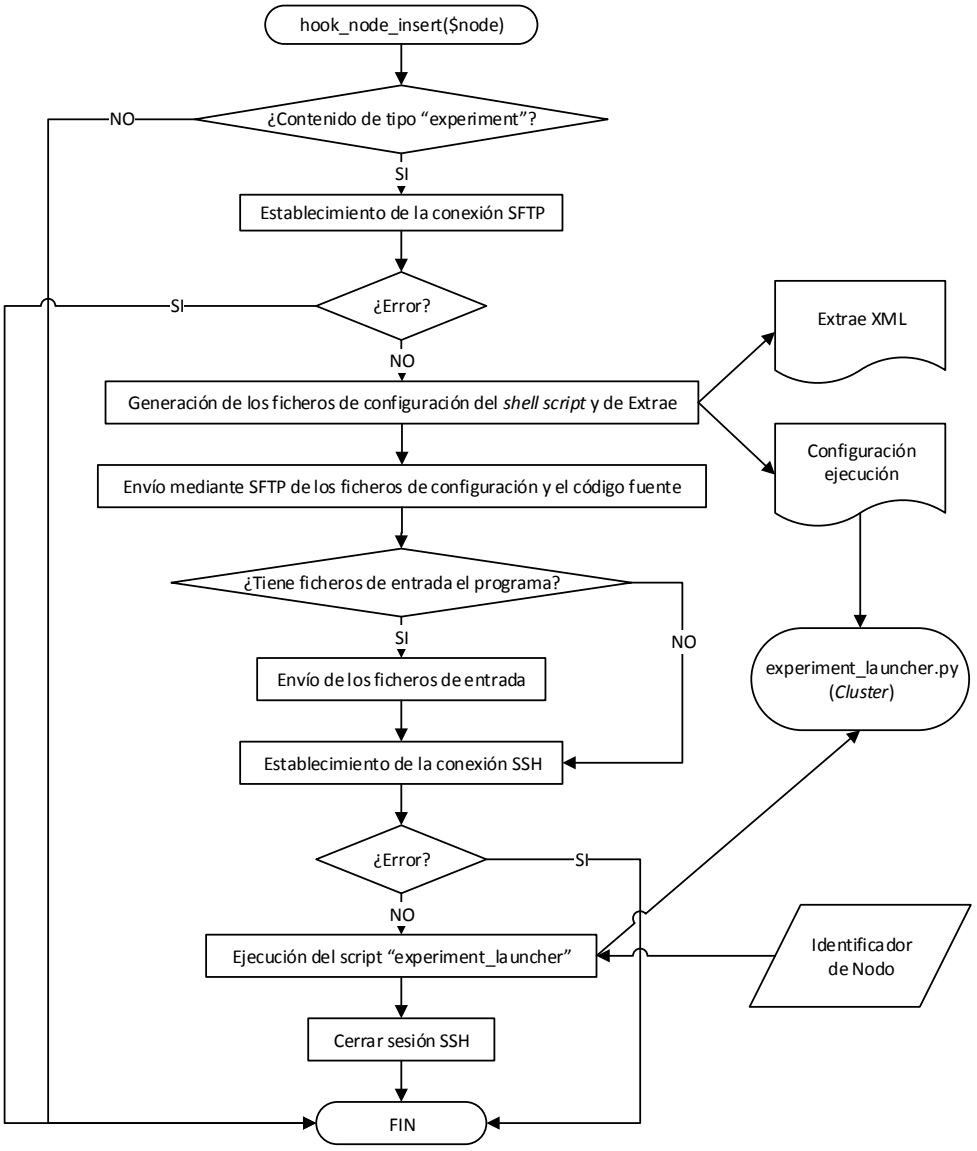


Figura 4.13: Driagrama de flujo del proceso de lanzamiento de un experimento.

4.1.6 Selección de una licencia asociada a la utilización de la traza (RF9)

Se han escogido las siete licencias de Creative Commons [11] como las licencias disponibles. Esta elección se ha llevado a cabo basándose en que son ampliamente usadas en la protección de datos, dado que se basan en cuatro condiciones que combinadas entre sí forman seis tipos distintos de licencia.

- Reconocimiento o *Attribution*, indica que en la explotación de la obra acogida a esta licencia se ha de reconocer su autoría. Habitualmente es identificada por el término "BY".
- No Comercial o *Non Commercial*, alude a que la explotación de la obra esta limitada a propósitos no comerciales. Esta condición es representada por el término "NC".
- Sin Obras Derivadas o *No Derivate Works*, indica que no se puede hacer uso de la obra si la transformación de ella se emplea para crear una obra derivada. El término para esta condición es "ND".
- Compartir Igual o *Share Alike*, que indica que se pueden crear obras derivadas siempre que la nueva obra mantenga la misma licencia.

Combinando estas condiciones se generan las siguientes licencias Creative Commons:

1. Reconocimiento (CC BY).
2. Reconocimiento - No Comercial (CC BY-NC)
3. Reconocimiento - No Comercial - Compartir Igual (CC BY-NC-SA)
4. Reconocimiento - No Comercial - Sin Obra Derivada (CC BY-NC-ND)
5. Reconocimiento - Compartir Igual (CC BY-SA)
6. Reconocimiento - Sin Obra Derivada (CC BY-ND)

La séptima y última licencia se denomina Creative Commons Zero (CC0), que indica que el autor de la obra renuncia a sus derechos patrimoniales y de autor sobre la obra en todo el mundo.

Para que el fichero de licencia correspondiente se encuentre junto a los ficheros ficheros de trazas, se ha incluido en el formulario de creación de experimentos un campo para la selección de la licencia, como se ha podido comprobar en el apartado anterior. Este campo será de tipo lista, y en el que se han definido siete entradas correspondientes a las siete licencias elegidas. Estas entradas están compuestas por dos valores, el nombre de la entrada que se mostrará al usuario en el formulario, y el correspondiente valor que se almacenará en el campo. De esta forma, asignando a esté último valor el nombre del fichero de la licencia se conocerá el fichero que hay que copiar junto con los ficheros de la traza. Estas correspondencia se realiza desde la interfaz web de administración destinada a la definición de campos, por lo que la adición o edición de las licencias disponibles es tan simple como modificar las entradas de este campo y añadir los ficheros de las licencias a los *clusters*.

4.1.7 Descarga de trazas (RF11)

Otro aspecto que el servidor web ha de integrar es la descarga de las trazas ya completadas. Dado que en algunos *clusters* no es posible establecer comunicaciones bidireccionales, es decir, acceder desde un equipo externo al sistema y viceversa, como es el caso de Altamira, la posibilidad de implementar la monitorización de la traza desde el propio *cluster* queda excluida. Por ello ha sido necesario recurrir a técnica conocida como *pooling*, aunque no sea la más recomendable en los casos en que sí se da esta condición.

Para implementar esta técnica y solventar este requisito funcional se ha seguido el procedimiento del diagrama de flujo representado en 4.14.

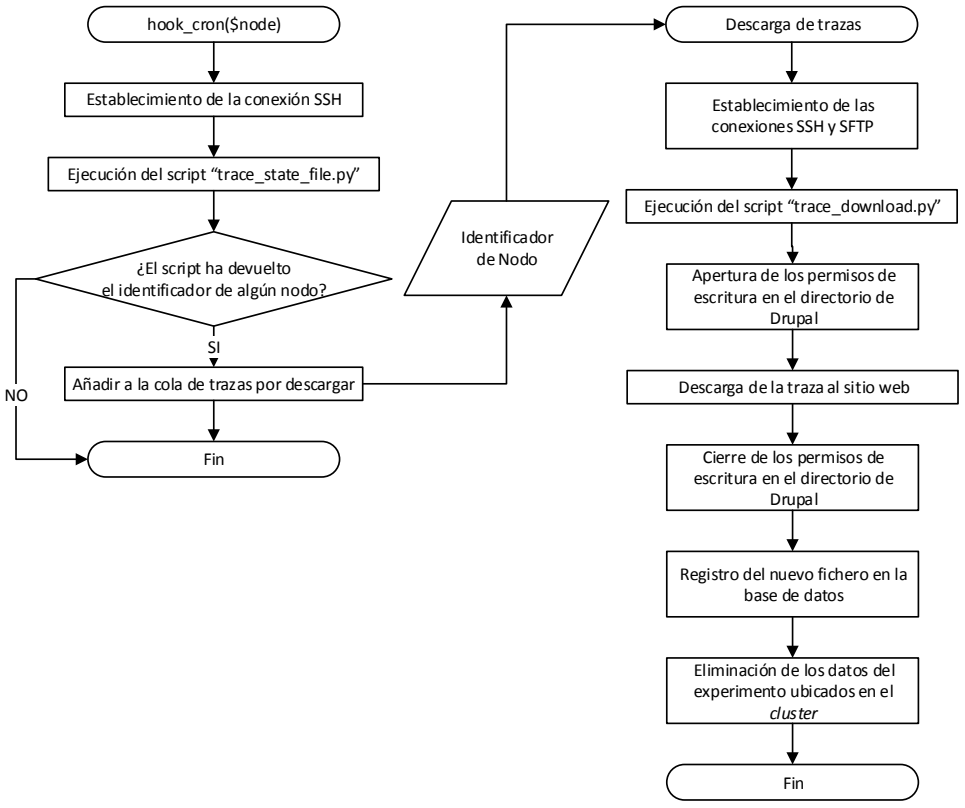


Figura 4.14: Diagrama de flujo de la implementación de la descarga de trazas en el servidor web.

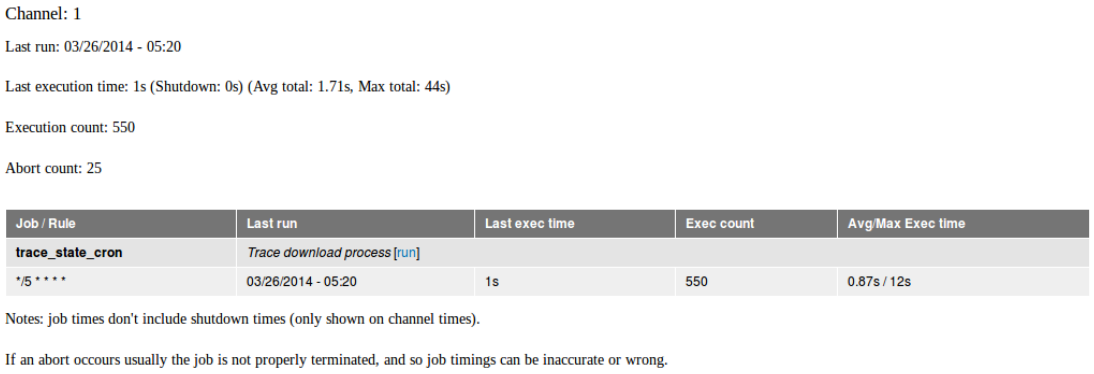
Dicho procedimiento comienza por el uso del *hook* "hook_cron($node)", que será invocado por Drupal transcurrido cierto intervalo de tiempo desde su última invocación. Es decir se trata de una función que será invocada periódicamente y cuyo periodo puede ser definido para ajustarse a las necesidades y la carga de trabajo del proceso que implementa. La configuración de este periodo también requiere de una configuración previa del "cron" del sistema operativo sobre el que se ejecuta el servidor web. Este *daemon* o proceso residente se encargará de ejecutar la funcionalidad de Drupal encargada de invocar a las tareas periódicas definidas en el sitio web. Por esta razón el intervalo de tiempo definido en el "cron" del sistema operativo ha de ser menor o igual que el tiempo deseado para la tarea de Drupal. En [42] se discute sobre la configuración de este proceso. En este caso se ha fijado este periodo en cinco minutos y se ha utilizado el

módulo “Elysia cron” [31].

El uso de este módulo, “Elysia Cron”, no es estrictamente necesario para lograr el propósito de ejecutar esta función de forma periódica, pero proporciona una serie de ventajas y facilidades al diseño. “Elysia Cron” permite administrar las tareas a ejecutar periódicamente desde una interfaz accesible desde el menú de administración de Drupal que se muestra en la figura 4.15. Con esta interfaz se pueden crear distintos canales en los que agrupar las tareas a ejecutar para especificar cuales se ejecutarán de forma simultanea, configurar los periodos, horas, o fechas de ejecución así como añadir *scripts* a ejecutar.

Por otra parte es posible emplear la API que pone a disposición este módulo para crear tareas a ejecutar de forma cronológica. En el caso de este proyecto se ha hecho uso de esta API para definir la función descrita en el diagrama de flujo de la figura 4.14 como una de estas tareas.

El principal motivo de uso de este módulo es la flexibilidad que ofrece al administrador del contenido para configurar los tiempos entre ejecuciones, así como la posibilidad de separar las tareas de mantenimiento de este proceso que es fundamental para el correcto funcionamiento del sistema.


Channel: 1

Last run: 03/26/2014 - 05:20

Last execution time: 1s (Shutdown: 0s) (Avg total: 1.71s, Max total: 44s)

Execution count: 550

Abort count: 25

Job / Rule	Last run	Last exec time	Exec count	Avg/Max Exec time
trace_state_cron	*Trace download process* [run]			
*/5 * * * *	03/26/2014 - 05:20	1s	550	0.87s / 12s

Notes: job times don't include shutdown times (only shown on channel times).

If an abort occours usually the job is not properly terminated, and so job timings can be inaccurate or wrong.

Figura 4.15: Fragmento de la página de administración del módulo "Elysia Cron".

De esta forma cada cinco minutos se ejecutará el *script* “trace_state_file.py” en el *cluster* mediante una sesión SSH. Este *script* se encargará de comprobar qué experimentos han completado su ejecución y retornará por la salida estándar los valores de los distintos identificadores de nodo asociados a dichos experimentos. Posteriormente, en caso de que “trace_state_file.py” devuelva algún identificador de nodo, se añadirán cada uno de los nodos correspondientes a una cola de trazas pendientes de descarga. Cada uno de los nodos que estén dentro de esta cola irán siendo procesados según la carga del sistema, evitando de esta forma problemas de sobrecarga cuando varias trazas estén listas para su descarga, dado el gran tamaño que pueden llegar a alcanzar estás. Para procesar estos nodos primero se ejecutará el *script* “trace_download.py” que requerirá como argumento de entrada el identificador del nodo, que se encargará de comprimir todos los ficheros correspondientes al experimento en un único archivo ZIP, para así facilitar la descarga. Por motivos de seguridad el directorio en el que se almacenarán estos ficheros estará protegido contra escritura. Posteriormente mediante el protocolo SFTP se descargarán las trazas al servidor web, habiendo abierto los permisos de escritura del directorio. Una vez la traza haya sido descargada se registrará en la base de datos del sitio web, y se actualizarán los registros del experimento correspondientes a la traza. De esta forma cuando un usuario acceda al repositorio de trazas, el campo asociado a los ficheros de la traza será actualizado, a los cuales podrá acceder.

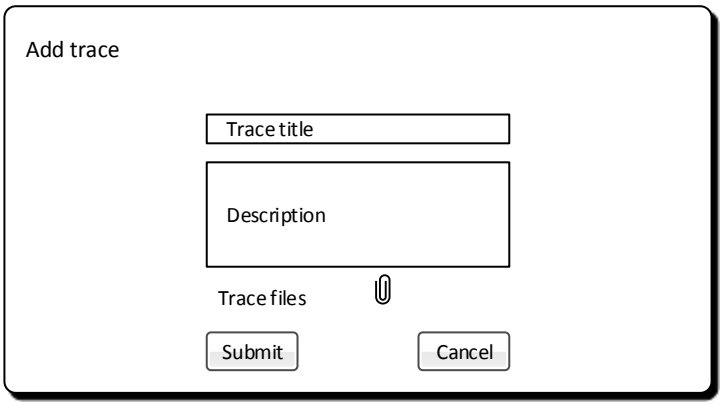


Figura 4.16: Formulario de creación del contenido de tipo “Trace”.

4.1.8 Subida de trazas (RF12)

Para lograr el propósito de que los usuarios puedan enviar sus trazas para que sean publicadas en el repositorio de trazas, se ha creado un nuevo tipo de contenido al que se le ha dado el nombre de “Trace”.

Este contenido está compuesto por tres campos como se refleja en el formulario representado en la figura 4.16.

1. *Title*. Nombre del nodo.
2. *Description*. Los comentarios que quiera hacer el autor sobre la traza que permitan describir en que consiste y que se van a encontrar el resto de usuarios al descargar los ficheros.
3. *Trace files*. Ficheros de la traza. En este campo se añadirá los ficheros de la traza comprimidos en formato ZIP.

Como se especifico en el requisito funcional, este contenido solo será publicado por administrador salvo que éste haya autorizado al usuario a publicar trazas. Para lograr este propósito se ha recurrido a un módulo llamado “Field Permissions” [34] que permite indicar que campos pueden editar los usuarios dependiendo de su rol dentro del sistema. De esta forma se ha definido un nuevo campo asociado a los usuarios denominado como “Trace Content” que únicamente será modificable por el administrador. Además se ha utilizado el “hook” “hook_node_presave($node)” para determinar sí el usuario tiene la autorización del administrador para publicar automáticamente sus trazas en el repositorio y proceder a ello en caso afirmativo como se recoge en el diagrama de flujo representado en la figura 4.17.

4.1.9 Listado de trazas (RF13)

Para mostrar el conjunto de trazas se ha empleado de nuevo el módulo “views”. En este caso se mostrarán los contenidos de tipo “Experiment” y “Trace” con los siguientes campos que se listan a continuación.

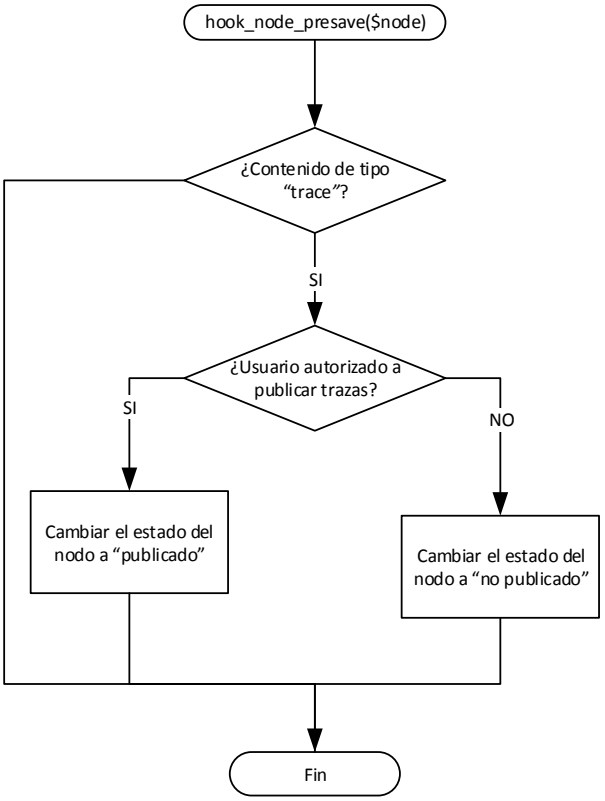


Figura 4.17: Diagrama de flujo para la publicación de las trazas.

- Título (Title).
- Nombre del autor (Author).
- Título del contenido de tipo "Source Code" empleado (Source Code Title).
- Nombre del *cluster* (Cluster).
- Fichero de la traza (Trace).
- Enlace al formulario para añadir citas (Add citation).

Como se puede observar en el fragmento de la página de configuración de la "view" en la figura 4.18, además de estos campos se han definido otros dos, "Source Code ID" y "Author ID".

El primero será empleado por el campo "Source Code Title" para obtener el título del nodo correspondiente. Al igual que ocurría con el campo "run experiment" de la "view" "Source Code List" se ha definido usando la API del módulo "views".

El segundo, "Author ID", se ha empleado para determinar si se ha de mostrar el campo "Add citation" como se verá en la implementación del requisito funcional RF14.

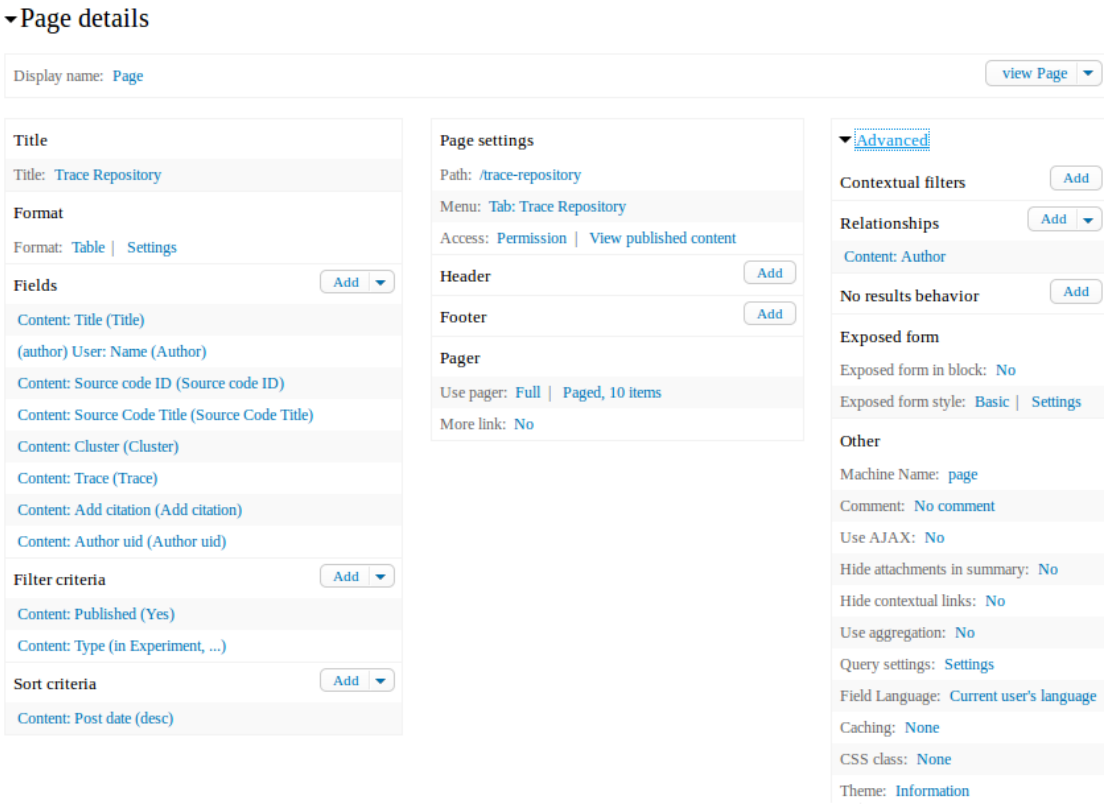


Figura 4.18: Página de administración de la "view" "Trace Repository".

4.1.10 Adición de citas a las trazas (RF14)

Para resolver el problema de añadir citas a los experimentos una vez estos han sido empleados en la creación de algún tipo de trabajo científico se ha recurrido a la API del módulo "views", al sistema de menús [16] y a la API de generación de formularios [15] de Drupal.

La API de "views" ha sido utilizada para definir un nuevo tipo de campo a la "view" creada para listar los experimentos. Este campo consiste en un enlace a la página del formulario que permitirá introducir la cita y enviarla para añadirla a los archivos del experimento.

El sistema de menús ha sido empleado para crear una página en la que se ubicará un formulario en el que introducir la cita. De esta forma el campo "Add citation" enlazará con esta página enviando el identificador de nodo del experimento al que se desea añadir la cita por medio de nuevo de la variable predefinida de PHP $\_GET$, que permite transferir información por medio de la URL.

La API de generación de formularios se ha usado para definir los campos del formulario y para introducir el proceso de añadir la cita al fichero CITATION.txt junto con el resto de archivos del experimento. Los campos serán dos, un campo de texto en el que el usuario introducirá la información que crea conveniente para que su obra sea citada, y un botón con el que enviar el contenido del formulario al sistema. El proceso que realizará el sistema consistirá simplemente en obtener la ruta al fichero de trazas correspondiente al experimento (se recuerda que es un fichero ZIP), y utilizando el método "addfromString" de la clase "ZipArchive" de PHP, agregar el contenido del campo de texto al archivo ZIP como un fichero de nombre CITATION.txt.

4.2 Implementación de la Funcionalidad en el *Cluster*

La estrategia llevada a cabo en el *cluster* se basá principalmente en dos ideas: organizar sistematicamente los directorios, y disponer una serie de *scripts* para la ejecución de las tareas. En el caso de Altamira se ha optado por utilizar el lenguaje de programación interpretado Pyhton [22] por ser un lenguaje que se adapta bien a las necesidades de la aplicación, al tratarse de un lenguaje multiparadigma. En este caso se usa ampliamente los estilos de programación orientada a objetos y de programación funcional.

El diseño que se recoge a continuación es la implementación en un *cluster* determinado, Altamira, pudiendo sufrir cambios en otros sistemas. Aún así las ideas recogidas son validas para cualquier sistema.

4.2.1 Estructura del directorio

La estructura que se ha seguido esta ilustrada en la figura 4.19. El propósito de esta estructura es establecer la ubicación de los distintos elementos necesarios y que se irán generando a medida que avanza el proceso de captura de trazas, para que el servidor web pueda comunicarse correctamente con cualquier *cluster*. En el directorio principal "trace_extraction" se identifican seis directorios.

- El subdirectorio "license" almacenará las distintos ficheros correspondientes a los siete tipos de licencias Creative Commons disponibles en la plataforma. La licencia seleccionada por el usuario en experimento será archivada junto sus ficheros de trazas para su posterior almacenamiento en el servidor web.
- En subdirectorio "log" se utilizara para registrar las salidas de error y estándar de la compilación y lanzamiento del experimento. De esta forma en caso de que la aplicación no funcione correctamente se pueden consultar estos ficheros que serán reconocibles por el identificar de nodo (nid) correspondiente al experimento.
- En el subdirectorio "scripts" se encuentran los *script* necesarios para implementar la funcionalidad, como se estudirá en los siguientes subapartados: "experiment_launcher.py", "trace_download.py", "trace_state_file.py" y "clean.py". El fichero "Makefile" será un archivo que permitirá la compilación de forma genérica de los códigos fuente escritos usando el lenguage de programación C con el empleo de la herramienta "GNU Make" [28]. README.txt, consiste en un fichero de texto plano a modo de plantilla utilizado para crear un fichero README.txt final que irá archivado junto con los ficheros de traza de cada experimento, y que estará personalizado indicando el autor, la fecha de creación, el *cluster* en el que se ha lanzado el experimento, e información referente al fichero LICENSE.txt. Por último, "job_submit.sh" será un fichero plantilla con el que se generará el *script* de lanzamiento del gestor de colas del *cluster* visto en el apartado 2.4.
- El siguiente subdirectorio, "node", contiene los datos necesarios para realizar los experimentos enviados desde el servidor web. Estos datos serán el archivo de configuración de la ejecución "nid.cfg", el fichero de configuración de Extrae "nid.xml", el código fuente "src.zip" y los distintos archivos de entrada del programa. Estos archivos estarán agrupados en directorios independientes para cada experimento. En todos estos ficheros y directorios "nid" significa identificador de nodo, y será el nombre que tome cada uno de ellos

para un nodo en concreto. Es decir un experimento que tenga asociado un identificador de nodo igual a diez, tendrá de nombre de directorio 10, 10.cfg como fichero de configuración y 10.xml como fichero de configuración de Extrae.

- El el directorio "src" se almacenará el código fuente descomprimido, y será el directorio de compilación y ejecución.
- En el directorio "trace" se almacenarán las trazas generadas y se copiarán los archivos adicionales como la licencia, los ficheros de salida, etc, y se comprimirá todo ello para generar lo que será el fichero de trazas que se descargará al servidor web. Estos dos últimos subdirectorios tendrán una estructura similar a "node" separando los archivos de cada experimento en directorios de nombre el identificador de nodo de cada experimento.
- Por último se encuentra el fichero "trace_state". En este fichero se registrarán los experimentos lanzados al *cluster* y se asociarán los identificadores de nodo (nid) con el identificador de trabajo (jid) asignado por el gestor de colas al lanzar la ejecución, es decir se guardará la correspondencia entre el experimento en el servidor web y el experimento en el *cluster*. Estas entradas se eliminarán cuando la traza haya sido transferida al servidor web.

4.2.2 experiment_launcher.py.

El *script* "experiment_launcher.py" cubre el requisito funcional "compilación y ejecución del código fuente" (RF10), y tiene como objetivo lanzar los experimentos de extracción de trazas al gestor de colas del *cluster*. En la figura 4.20 se representa el diagrama de flujo de la implementación. Este *script* tiene un argumento de entrada, el identificador de nodo del experimento correspondiente (nid). Con ese parámetro se podrá leer el archivo de configuración correspondiente almacenado en "nodes/nid/" donde "nid" será el valor correspondiente. Después se extraerán todos los archivos del código fuente en "src/nid/", se copiará el "Makefile" y se compilará. Posteriormente se reemplazarán los parámetros obtenidos del fichero de configuración en la plantilla del *script* de lanzamiento. Con este *script* será posible el envío del trabajo a la cola de tareas. Con la salida generada por el comando de lanzamiento, se obtendrá el identificador de trabajo asociado al experimento. Este valor identificará al experimento dentro del gestor de colas, por lo es de vital importancia conocerlo y almacenarlo en el fichero "trace_state". Para evitar problemas de concurrencia entre la escritura de este fichero cuando este *script* es ejecutado para añadir una entrada en el fichero y cuando el *script* "trace_state_file.py" es lanzado para eliminar la entrada correspondiente a la traza transferida, se ha definido una nueva clase de objeto que permite bloquear y desbloquear el fichero. Esta clase esta basada en la excepción que se produce en Python al intentar crear un fichero que ya existe al usar la opción "os.O_CREAT" del método "open" de la clase "os". Con el fichero bloqueado será posible añadir la entrada "nid" - "jid". Hecho esto se liberará el fichero "trace_state" completando el procedimiento.

4.2.3 trace_state_file.py

El propósito de este *script* es el de controlar que experimentos han finalizado y cuales siguen en ejecución en el *cluster*. Para implementar esta funcionalidad se ha recurrido al procedimiento

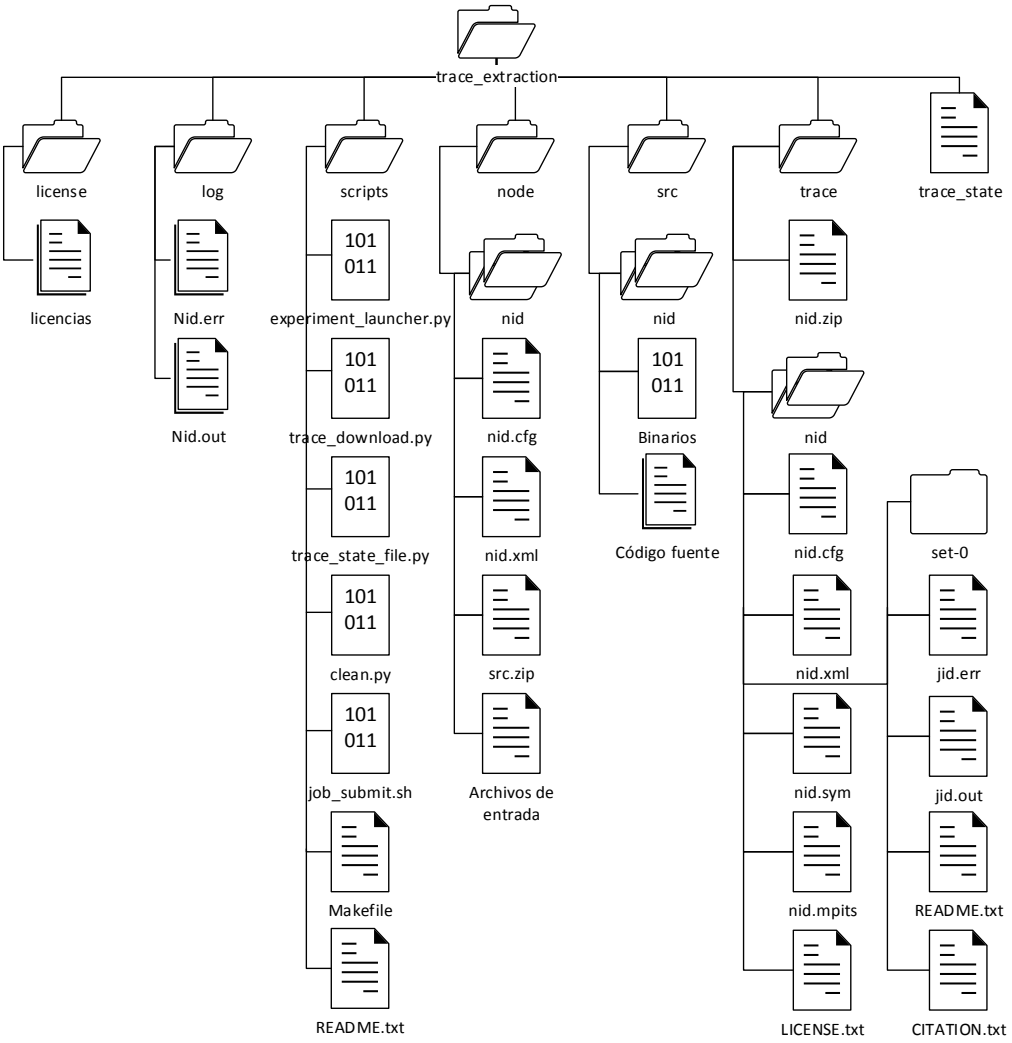


Figura 4.19: Estrucutura del directorio de la plataforma.

definido en el diagrama representado en la figura 4.21. Este proceso comienza con la lectura del fichero “trace_state” en el que se recogen las correspondencias “nid” - “jid” (identificador de nodo del experimento en Drupal con el identificador del trabajo correspondiente el *cluster*). El siguiente paso consiste en capturar la salida obtenida de la ejecución del comando de monitorización de los trabajos en el gestor de colas. Posteriormente se hará una búsqueda de los “jid” en la salida obtenida con el comando, en caso de que un “jid” no se encuentre en esta salida significará que el trabajo ha finalizado. De esta forma se imprimirán por la salida estándar los identificadores de nodo (nid) de aquellos trabajos finalizados, indicando a Drupal que puede comenzar la transferencia de los ficheros de la traza correspondiente. Además se eliminará las entradas correspondientes del fichero “trace_state” por los motivos ya comentados en el procedimiento del *script* “experiment_launcher.py”.

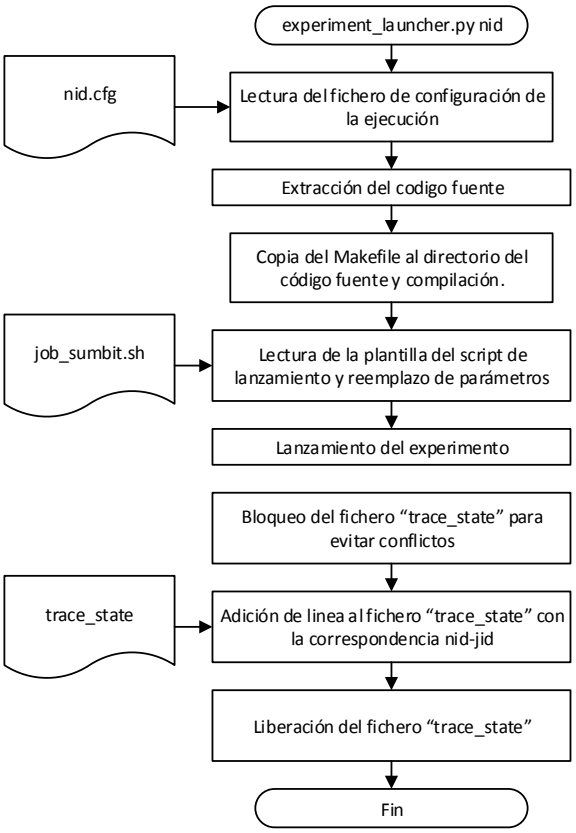


Figura 4.20: Diagrama de flujo correspondiente al *script* "experiment_launcher.py"

4.2.4 trace_download.py.

Este *script*, cubre el requerimiento funcional “transferencia de la traza” (RF11), y se encargará de comprobar que trazas están completas, y de recoger todos los archivos asociados al experimento para su posterior envío al servidor web.

Este *script* también tiene como argumento de entrada el identificador de nodo correspondiente al experimento, como se refleja en el diagrama de flujo de la figura 4.22. Con este identificador podrá acceder al fichero de configuración correspondiente y así determinar qué ficheros ha de copiar al directorio donde se ha generado la traza. Estos ficheros serán la salida estándar y de error en caso de que el usuario lo desee, y la licencia elegida. Después se realizará la conversión de las trazas al formato PRV usado por la herramienta Paraver, y por último se archivará todo en un fichero ZIP para facilitar la descargar y el registro de los ficheros de la traza en la base de datos del servidor web.

4.2.5 clean.py

El propósito de este *script* es eliminar todos los archivos relacionados con el experimento una vez que los ficheros de la traza estén alojados en el servidor web. De esta forma no se

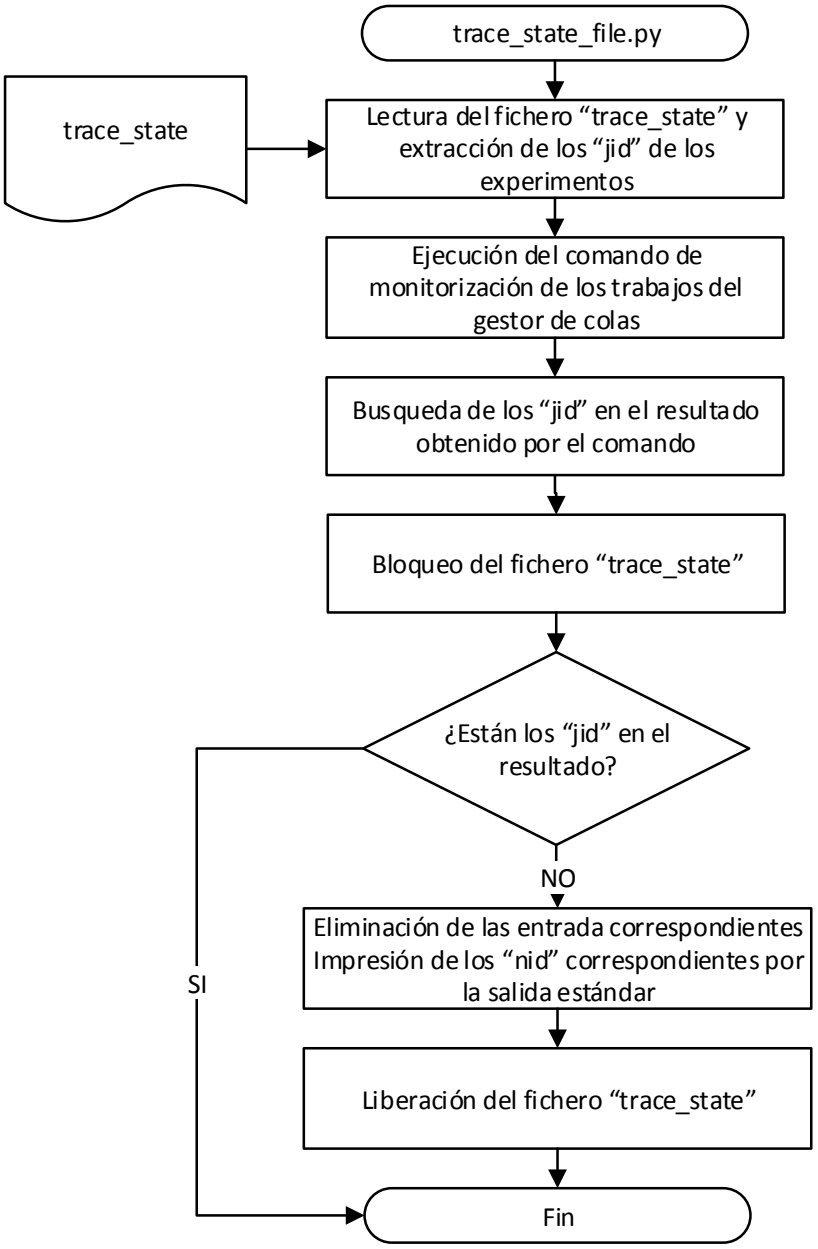


Figura 4.21: Diagrama de flujo del proceso de eliminación de entradas del fichero “trace_state”.

mantendrá información innecesaria en el *cluster* y se evitarán posibles problemas con las cuotas del sistema de ficheros. En la figura 4.23 se ilustra el diagrama de flujo de su implementación. Este *script* será ejecutado cuando la descarga de una traza se haya completado, como se aprecia en la figura 4.14, y, necesita el identificador de nodo del experimento (nid) como parámetro de entrada.

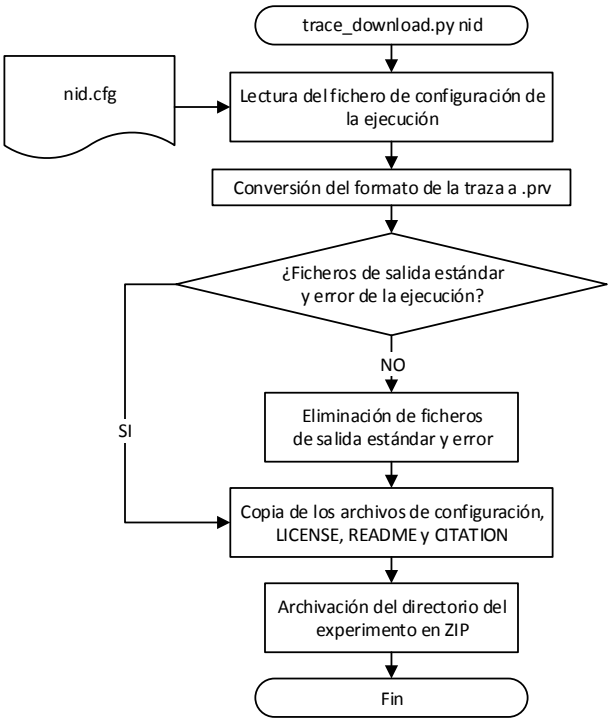


Figura 4.22: Diagrama de flujo de la funcionalidad del *script* "trace_download.py"

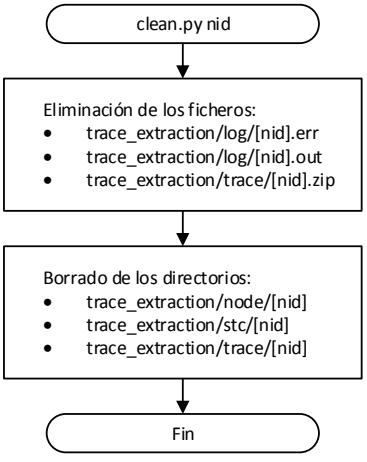


Figura 4.23: Diagrama de flujo de la funcionalidad del *script* "clean.py".

4.3 Conceptos necesarios para añadir un nuevo *cluster*

Como se ha podido ver en el desarrollo del sistema se han utilizado una serie de mecanismos para transferir la información desde el servidor web al *cluster* y viceversa. En este apartado se pretende explicar detalladamente estos mecanismos con objeto de explicar cómo se ha de proceder para incluir un nuevo *cluster* al sistema.

En primer lugar se puede apreciar que todos los *scripts* ejecutados en el *cluster* tienen como parámetro de entrada el identificador del nodo (nid) correspondiente al contenido de tipo "Experiment" generado.

Además, junto con los ficheros del código fuente, también se envía un fichero de configuración con la extensión CFG. En este fichero se recoge toda la información necesaria para poder ejecutar el programa enviado y generar los ficheros correspondientes del experimento. Este fichero tiene la estructura que se muestra en el código 4.1. Los *script* que se incluyan han de ser capaces de interpretar este tipo de ficheros ya que será el método por el que se transferirá toda la información necesaria desde Drupal a cualquier *cluster*.

```
[ExperimentProperties]
Title = Example
NodeID = 223
User = admin
Created = Fri, 03/21/2014 - 05:29
SourceCode = floyd.zip
TotalTasks = 10
TasksPerNode = 5
WallClockLimit = 01:00:00
[ProgramArguments]
Arguments = matrix_1000.txt
ProgramFiles = matrix_1000.zip
[Other]
Cluster = Altamira
ExtraeOutputAndError = 1
License = CC-by-nc-sa.txt
```

Código 4.1: Ejemplo de fichero de configuración.

En el ejemplo se distinguen dos tipos de líneas. Por un lado las propiedades que están compuestas por una clave y un valor delimitados por el signo igual, "nombre de la clave = valor". Estas propiedades se corresponden con los datos almacenados en el fichero. Por el otro las secciones escritas entre corchetes, "[Nombre de la sección]", que permiten organizar los datos, permitiendo una programación limpia y legible. Esta estructura es habitual en el entorno *software* por lo que ya existen funciones que permiten leer los datos almacenados en este tipo de ficheros. En Python, el lenguaje de programación empleado para desarrollar los *script* en Altamira, esta funcionalidad la provee la biblioteca ConfigParser [23].

El sistema podrá usar la información de este fichero para implementar la funcionalidad recogida en el apartado 4.2. Se utilizará los valores de "User", "Created" y "Custer" de las secciones "ExperimentProperties" y "Other" para personalizar el fichero README.txt y así identificar al autor, la fecha de creación del experimento y el *cluster* al que se ha lanzado. La propiedad "SourceCode" para identificar los archivos a descomprimir correspondientes al código fuente. "ProgramFiles" identifica al fichero ZIP en el que se almacenan los ficheros de entrada necesarios en la ejecución del programa, en caso de que los requiera. "TotalTasks", "TasksPerNode",

"WallClockLimit", y "Arguments" serán utilizados para generar el *script* de lanzamiento necesario por el gestor de colas. "ExtraeOutputAndError" indica si el usuario desea incluir los ficheros de error y salida estándar generados durante la ejecución del experimento. Con la propiedad "License" se determinará qué fichero de licencia será agregado a los archivos del experimento.

La otra parte en la que existe una comunicación entre el servidor web y el *cluster* se encuentra en el proceso de descarga de los archivos generados en los experimentos. Como se explicó en el apartado 4.2.1, en el fichero "trace_state" se recogen las correspondencias entre el identificador del experimento dentro del *cluster* (jid) y el identificador en Drupal (nid). De esta forma cuando Drupal ordene la ejecución del *script* "trace_download.py", en el *cluster* se procesará este archivo junto con el comando de verificación del estado del trabajo en el gestor de colas, en este caso "mnq", determinando qué experimentos han finalizado. Enviando el identificador de nodo (nid) de los experimentos que han finalizado por medio de la salida estándar, esta salida es leída por Drupal, se procede a la descarga de los ficheros ejecutando el proceso indicado en el apartado 4.1.7. La otra parte en la que existe una comunicación entre el servidor web y el *cluster* se encuentra en el proceso de descarga de los archivos generados en los experimentos. Como se explicó en el apartado 4.2.1, en el fichero "trace_state" se recogen las correspondencias entre el identificador del experimento dentro del *cluster* (jid) y el identificador en Drupal (nid). De esta forma cuando Drupal ordene la ejecución del *script* "trace_download.py", en el *cluster* se procesará este archivo junto con el comando de verificación del estado del trabajo en el gestor de colas, en este caso "mnq", determinando qué experimentos han finalizado. Enviando el identificador de nodo (nid) de los experimentos que han finalizado por medio de la salida estándar, esta salida es leída por Drupal, se procede a la descarga de los ficheros ejecutando el proceso indicado en el apartado 4.1.7.

En caso de añadir nuevos *cluster* también será necesario actualizar la implementación del servidor web. Para ello se propone el proceso detallado a continuación. En primer lugar será necesario crear una nueva tabla en la base de datos de Drupal. Esta tabla tendrá cuatro columnas: el nombre del *cluster*, su dirección IP, el nombre de la cuenta de usuario usada para acceder al *cluster* y la contraseña de este. En segundo lugar será necesario alterar el código fuente del módulo creado, comprobando la selección del *cluster* y accediendo a la base de datos para recoger la información almacenada en la nueva tabla y poder establecer las conexiones SSH y SFTP. En el caso del proceso de descarga de las trazas el cambio ha realizar puede ser algo más complicado dependiendo de la cantidad de *clusters* de los que se disponga, teniendo en cuenta que el tiempo transcurrido entre ejecuciones de la función definida en el apartado 4.1.7 es de cinco minutos. En caso de ser una cantidad reducida el problema se puede solventar accediendo a cada uno de los *clusters* de forma sucesiva y comprobando el estado de los experimentos en cada uno de ellos, disminuyendo la frecuencia entre ejecuciones de este proceso en caso de que sea necesario. En el caso de que la lista sea lo suficientemente grande como para que el tiempo de ejecución de este proceso de comprobación sea considerable, es posible que haya que emplear métodos más sofisticados que permitan llevar un control de los experimentos lanzados en el propio servidor web, unicamente accediendo a los *clusters* que tengan experimentos activos en ese instante de tiempo. Esto también se puede ver afectado por la carga que sufra el servidor web o el tipo de conexión que exista entre el servidor web y dichos *cluster* por lo que no es posible dar una solución formal a este tema. Este aspecto ha de ser tratado en función de estos factores por lo que en el hipotético caso de incluir algún *cluster* al sistema diseñado, se recomienda actuar conforme a ellos empleando técnicas de *profiling*.

CAPÍTULO 5

VERIFICACIÓN Y VALIDACIÓN DE LA APLICACIÓN

El motivo de este capítulo es verificar el funcionamiento de las características más importantes del sistema diseñado. Durante el proceso de desarrollo se realizaron una serie de pruebas unitarias comprobando cada elemento de forma específica, comprobando los entresijos de cada uno de ellos. Con el conjunto de elementos verificado individualmente, se procedió a realizar pruebas de integración, en las que se examinaba el comportamiento del sistema de manera conjunta.

En este capítulo se presenta un ejemplo en el que se verifica el sistema desde la creación de una cuenta de usuario hasta la realización de un experimento, pasando por las distintas fases que esto conlleva. Al final del mismo, aunque no sea parte de la herramienta diseñada, se muestran dos ejemplos de trazas con los que se puede apreciar la utilidad de esta herramienta.

5.1 Solicitud de cuenta de usuario en el sistema

El primer paso es acceder al servidor web. Una vez se haya completado la descarga de la página de inicio, el navegador web mostrará una página similar a la mostrada en la figura 5.1. En ella se distingue el menú de navegación en la parte superior derecha y en la parte superior izquierda, el menú de autenticación de usuarios en la parte inferior izquierda, los últimos contenidos publicados en la parte central, así como los comentarios y el contenido reciente en la parte derecha.

El siguiente paso es el de solicitar la cuenta de usuario. Para ello pulsando sobre la opción "Create new account" del menú de autenticación de usuario, se accederá a la página de creación de la cuenta de usuario mostrada en la figura 5.2. En ella se le solicita al usuario un nombre de usuario y una dirección de correo electrónico válida. Al pulsar sobre el botón "Create new account" la cuenta será creada pero seguirá teniendo las limitaciones de usuario anónimo hasta que el administrador la apruebe. En el ejemplo se emplea el nombre de usuario "user_example" que será utilizado a lo largo del ejemplo.

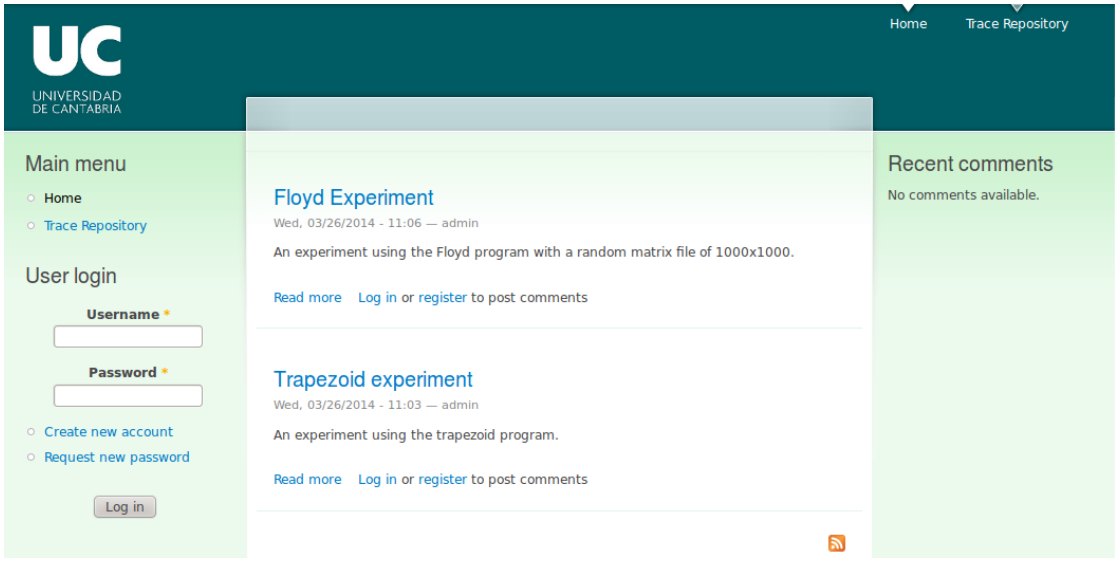


Figura 5.1: Página de inicio del sitio web.

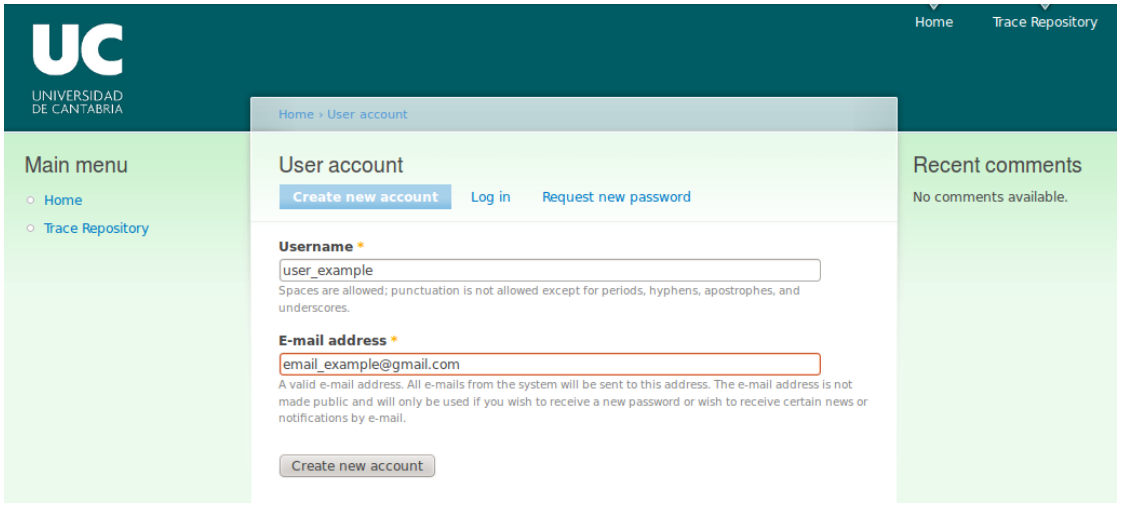


Figura 5.2: Página de creación de una nueva cuenta.

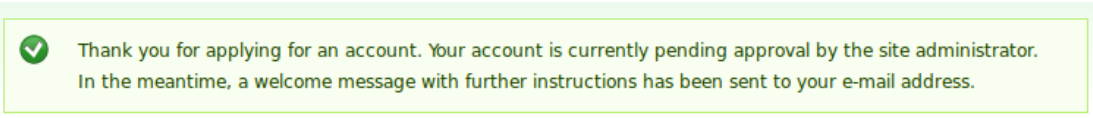


Figura 5.3: Mensaje enviado al usuario una vez completado el formulario

5.2 Activación de la cuenta de usuario

El administrador accederá a la página de gestión de usuarios, y si no encuentra ningún inconveniente autorizará la creación de la cuenta desbloqueandola como se muestra en la figura 5.4. Se aprecia cómo se ha seleccionado la opción “unblock the selected users” y se ha marcado la fila correspondiente al usuario “user_test”, creado en la fase anterior, por lo que al pulsar el botón “Update” se habilita esta cuenta.

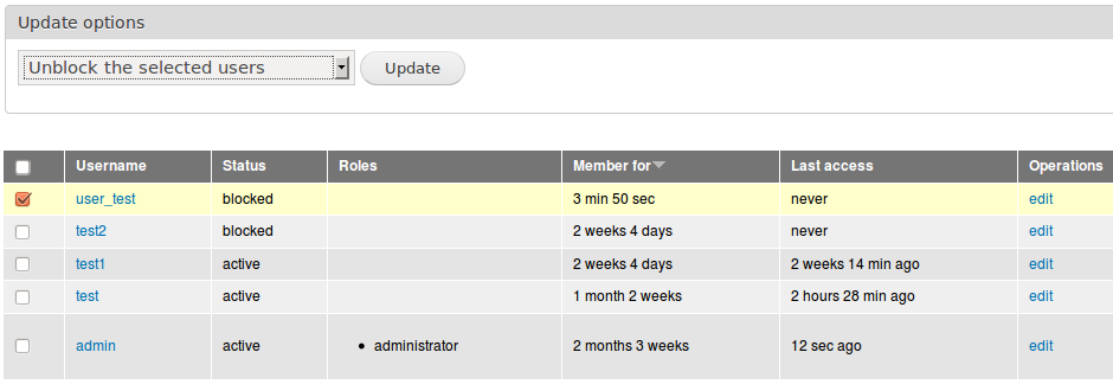


Figura 5.4: Página de administración de las cuentas de usuario

5.3 Accediendo al repositorio de trazas I

Accediendo al repositorio de trazas todavía como usuario anónimo se comprueba que cualquier usuario tiene total acceso a ellas pudiendo descargar los ficheros de la traza como se muestra en la figura 5.5. De esta forma se podrá comparar el estado de la lista antes y después de lanzar un experimento.

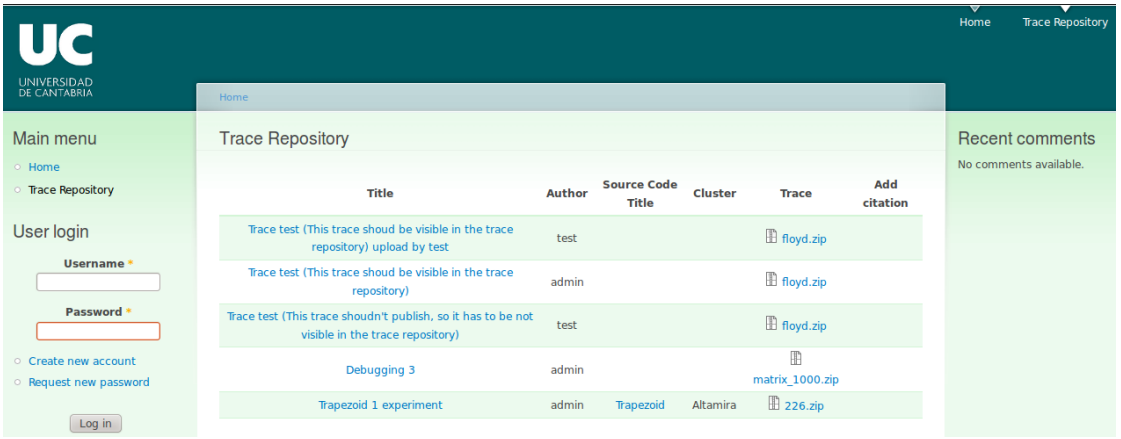


Figura 5.5: Página correspondiente al repositorio de trazas accediendo como usuario anónimo

.

5.4 Autenticación en el sistema

Para poder subir código fuente es necesario estar autenticado por ello este será el siguiente paso. Introduciendo los credenciales en los campos y pulsando el botón “Login” del menú de autenticación se accederá al sistema como un usuario registrado y serán visibles tres nuevas opciones, añadir código fuente (Add Source Code), realizar experimento (Run Experiment) y añadir traza (Add Trace), en el menú de navegación y será visible el menú de configuración de la cuenta de usuario, como se muestra en la figura 5.6.

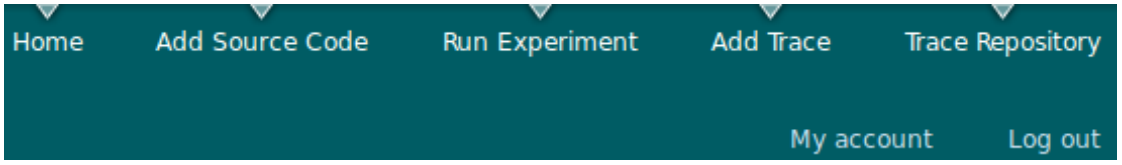


Figura 5.6: Menús de navegación y menú de configuración de la cuenta de usuario

.

5.5 Subiendo un código fuente

Para poder subir código es necesario rellenar los campos del formulario que se muestra en la figura 5.7. El nombre que se le ha dado a este contenido ha sido "user_test source code" que será empleado para realizar el experimento de ejemplo.

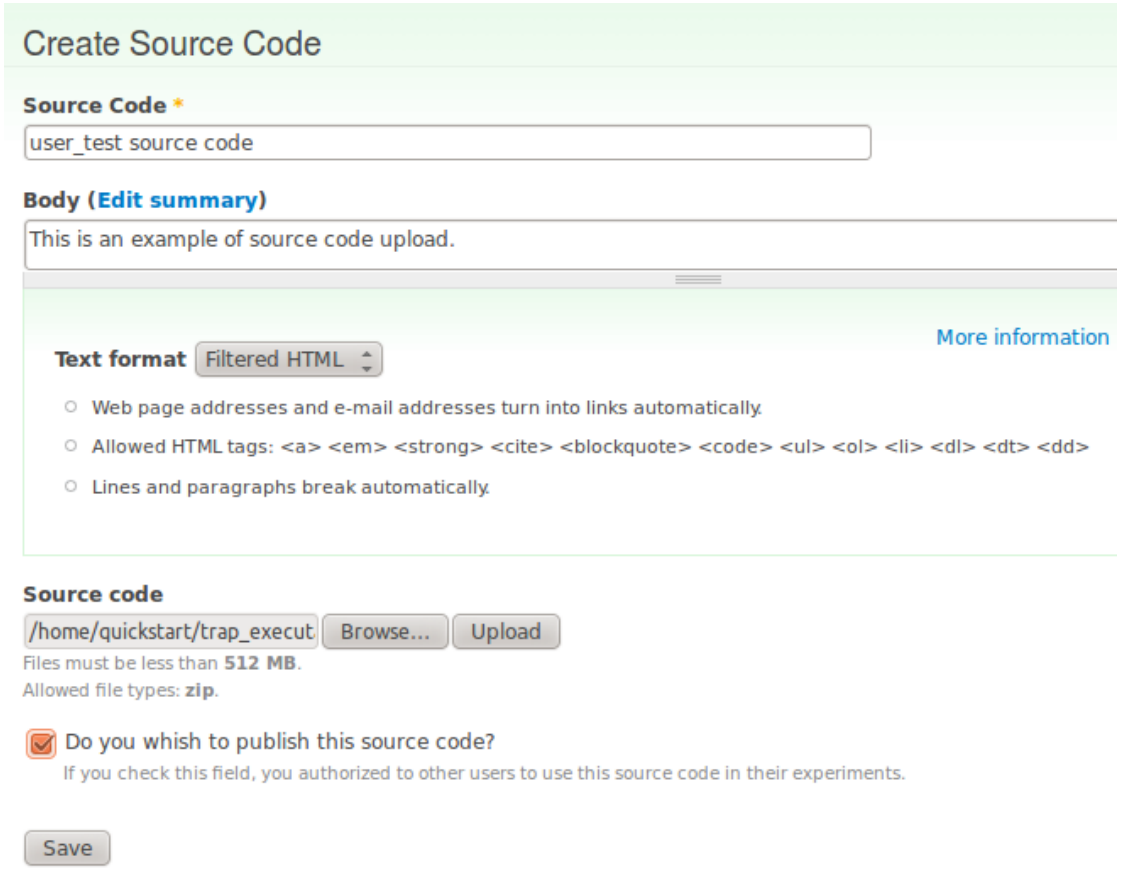


Figura 5.7: Página de creación de contenido de tipo código fuente.

5.6 Realización de experimentos

Para poder extraer las trazas de la ejecución de un programa es necesario crear un experimento. Para ello accediendo a la opción "Run Experiment" del menú de navegación se accederá al listado de códigos fuente disponibles como se representa en la figura 5.8.

Después presionando sobre uno de los códigos fuente, el usuario es dirigido al formulario de

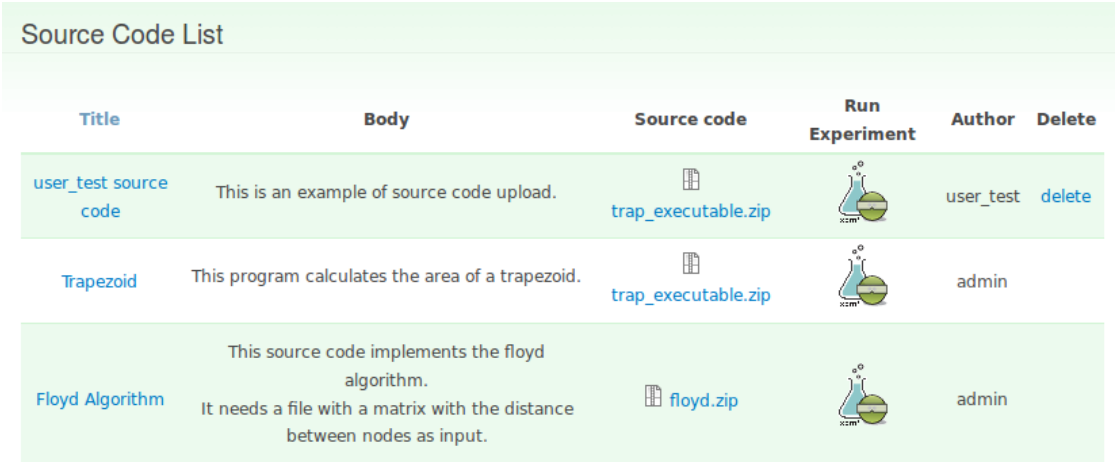


Figura 5.8: Página con la lista de códigos fuentes disponibles para su uso en la realización de experimentos.

creación de experimentos reflejado en la figura 5.9. En este caso se ha elegido el código fuente creado anteriormente que tiene por título “user_test source code”.

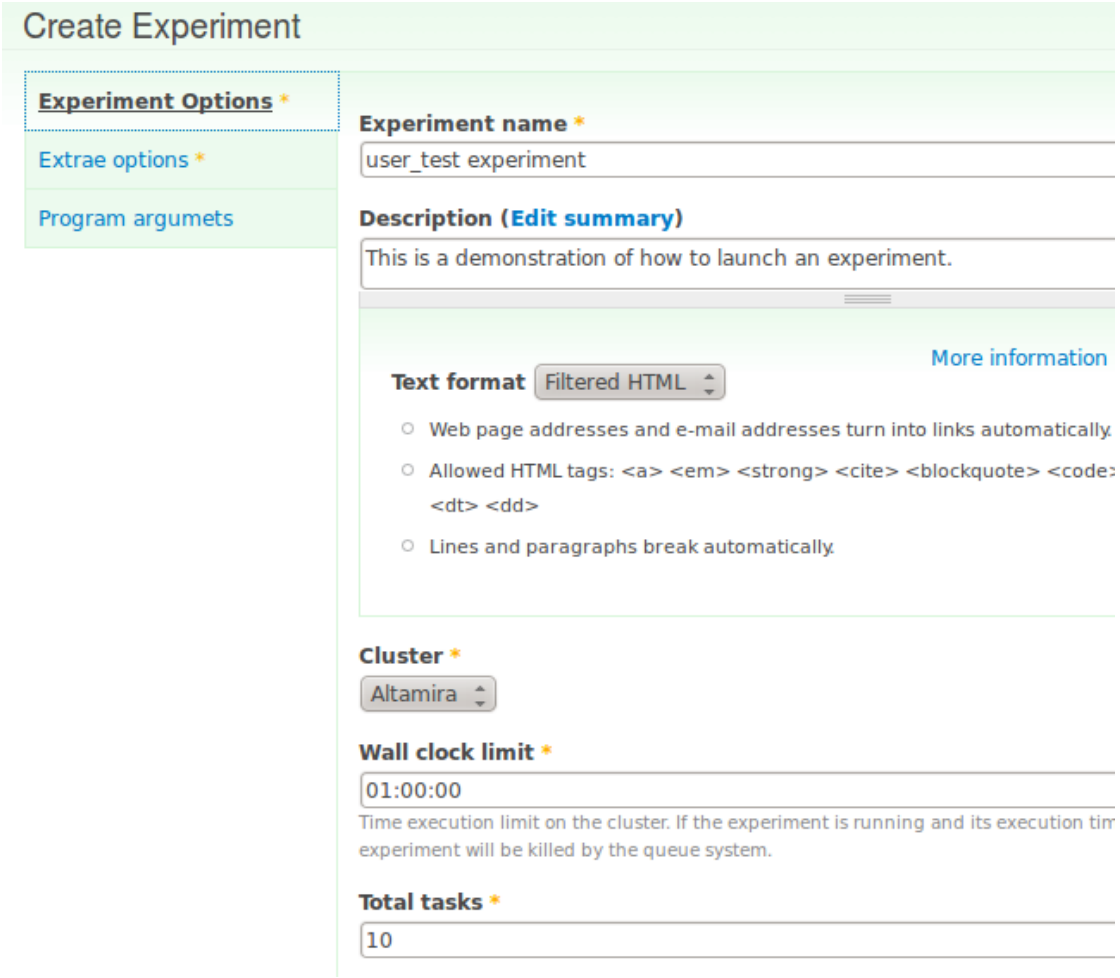


Figura 5.9: Fragmento de la página de creación de experimentos.

En esta figura no se muestran todos los campos que el usuario ha de rellenar pero a modo de

ejemplo basta con mencionar que el nombre que se le ha dado al experimento es el de “user_test experiment”.

5.7 Accediendo al repositorio de trazas II

Transcurrido el tiempo necesario para que se complete la ejecución en el *cluster* y la descarga de los ficheros de la traza al servidor web se accede de nuevo al repositorio de trazas. Observando la figura 5.10 se aprecia como el experimento creado tiene asociado un fichero en la columna “Trace”. Este fichero ZIP se corresponde con los archivos de la traza, licencia, etc. También se aprecia en las columnas “Author” y “Source Code ID” se encuentran los valores 3 y 230, que se corresponden con el identificador de usuario del autor del experimento y el identificador del nodo correspondiente al experimento creado. También en la columna “Add Citation” se observa como aparece un icono en la linea correspondiente al experimento creado por este usuario. Como se observa en el resto de filas no aparece este icono, y esto es debido a que esos experimentos no son de su autoría. Si el usuario pulsa sobre el icono será dirigido a la página de creación de la cita del experimento.

Figura 5.10: Página del repositorio de trazas. Experimento completado con su fichero de trazas correspondiente.

5.8 Adición de una cita

Accediendo al repositorio de trazas y pulsando sobre la opción “Add citation” de un experimento se accederá al formulario representado en 5.11. En el campo “Citation” el autor deberá escribir el texto con el que se citará al trabajo en el que ha sido empleada la traza seleccionada. Por defecto se propone una plantilla de la referencia bibliográfica que se usaría para citar un articulo con la herramienta BibTeX, un recurso ampliamente utilizado en la redacción de documentos utilizado para hacer referencia a otras obras.

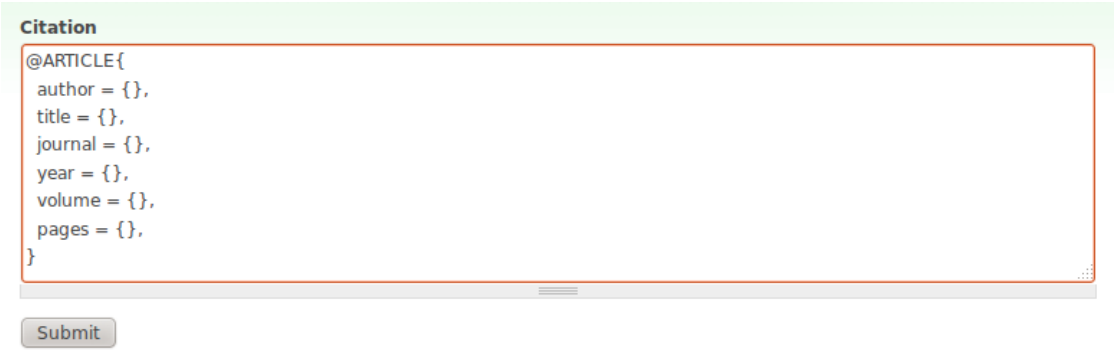


Figura 5.11: Formulario para la adición del fichero CITATION.txt al fichero de trazas del experimento.

5.9 Visualización de la traza

Aunque esto no forma parte de la herramienta construida, a través de estos ejemplos se puede dar una clara idea de los usos que pueden dar las trazas a los usuarios.

Una vez la traza ha sido descargada, el usuario podrá hacer uso de la herramienta "Paraver" para visualizar la información recogida en los ficheros de la traza.

En el ejemplo que se muestra en la figura 5.12 se representa un extracto de la traza obtenida en la ejecución de la aplicación paralela "CGPOP" [2]. Aunque la escala no sea la más adecuada para entrar en detalles, se pueden identificar algunos de los estados por los que pasa esta ejecución que consta de veinticuatro procesos. En esta imagen en concreto se pueden identificar tres fases en las que se encuentran los procesos. En azul se representan los procesos en fase de computo, haciendo uso de la CPU. En rojo se representan los procesos en fase de espera. En naranja las comunicaciones colectivas, comunicaciones entre varios procesos. Las lineas amarillas representan la comunicación entre dos procesos. Existen una gran cantidad de estados más y estos son solo algunos ejemplos de lo que el usuario se puede encontrar al analizar la traza.

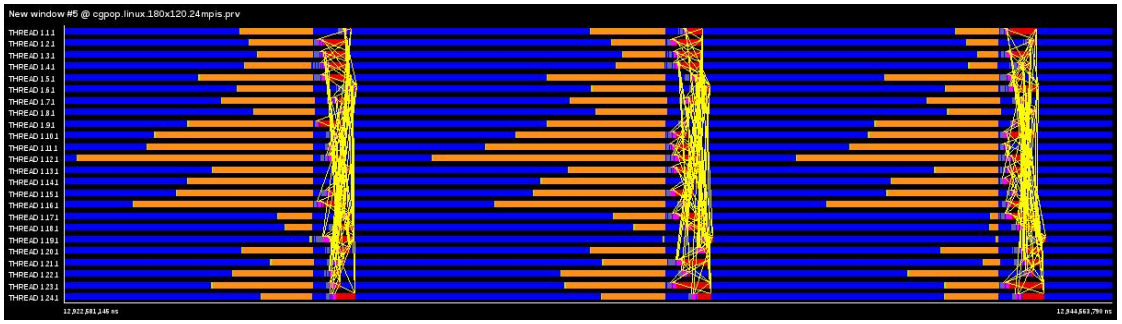


Figura 5.12: Fragmento de la traza obtenida de la ejecución en 24 procesos de la aplicación "CGPOP" visualizado con la herramienta "Paraver".

En este ejemplo se aprecian varias características de la ejecución. Una de ellas es la implementación de la programación paralela usando métodos de comunicación colectiva entre procesos, distinguiendo tres fases en el fragmento de la imagen. Además también es apreciable cómo la carga de trabajo entre los distintos procesos esta desequilibrada. En las tres fases de comunicación colectiva la duración de este estado en los distintos procesos es muy desigual con grandes periodos de espera para la mayoría de ellos debido a las grandes ráfagas de computo que sufre el proceso "THREAD 1.19.1". De esta forma, gracias a esta traza, se puede identificar un proceso

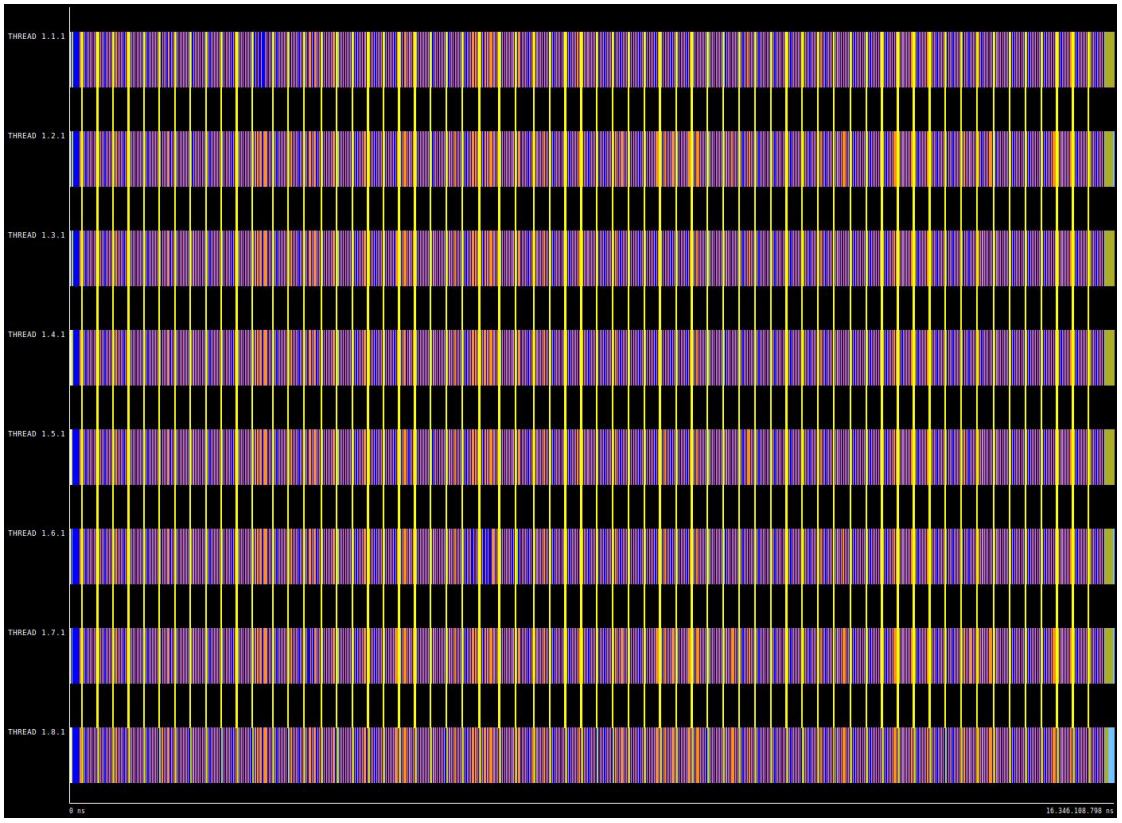


Figura 5.13: Fragmento de la traza obtenida de la ejecución en 8 procesos de la aplicación "Graph500" visualizado con la herramienta "Paraver".

de desbalanceo de carga, ayudando al programador a determinar si es necesario realizar alguna modificación en la aplicación.

Es posible encontrarse ejecuciones con características dispares a la anterior como es el caso de la traza presentada en la figura 5.13. Esta traza se corresponde con la ejecución de un *benchmark* del "Graph 500" [6]. En esta visualización se aprecia cómo en esta aplicación predomina la comunicación y es totalmente distinta a la de la aplicación "CGPOP" que se basaba en ráfagas de computo y fases de comunicación colectiva.

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

Con la finalización del proyecto se han obtenido una serie de conclusiones y se han contemplado posibles mejoras que pueden hacer de la plataforma desarrollada, una herramienta más productiva y de mayor valor.

6.1 Conclusiones

En la introducción de este documento, el capítulo 1, se definieron una serie de objetivos. Para cubrir estos objetivos se ha pasado por distintas fases, partiendo desde un proceso de aprendizaje, continuando con el establecimiento de una serie de requisitos, el planteamiento del diseño y la implementación, y por último la verificación del sistema. A continuación se analizan los distintos objetivos tanto los cumplidos como los no cumplidos.

- Se ha creado una plataforma web que permite a los usuarios enviar sus códigos fuente de aplicaciones paralelas para extraer trazas en el supercomputador Altamira, para posteriormente almacenar estos datos en un repositorio dentro de esta plataforma.
- Se ha provisto al usuario de una interfaz sencilla sin perder las posibilidades de configuración que tienen tanto el ajuste de los parámetros referentes a la ejecución de la aplicación, como los ajustes de Extrae. Para este último se ha creado una interfaz que permite establecer la configuración de la herramienta de tres formas diferentes para dar la posibilidad a los usuarios avanzados de sacar todo el provecho de Extrae y a los más noveles introducirles a la herramienta con un formulario simplificado.
- El diseño de la plataforma se ha realizado teniendo siempre en mente que su objetivo final es incluir varios sistemas de cómputo de alto rendimiento para proporcionar a los usuarios una amplia variedad de arquitecturas sobre las que ejecutar la extracción de trazas de sus aplicaciones. Dado que se ha trabajado sobre un único *cluster* no se ha verificado que el sistema sea completamente flexible.
- La plataforma dispone de un método con el que los usuarios pueden bloquear el acceso a sus códigos fuentes basado en el estado de publicación del contenido de Drupal, permitiendo al usuario decidir si publica o no el contenido. La implementación de este método evita que los usuarios puedan acceder al contenido privado de otros usuarios.

- Se ha hecho un amplio uso de las herramientas proporcionadas por Drupal diseñadas para gestionar los usuarios, para crear un sistema que requiera de una cuenta de acceso al servidor web para poder hacer uso del proceso de lanzamiento de experimentos con el fin de proteger los sistemas HPC.
- Los usuarios pueden elegir entre siete licencias de Creative Commons para proteger los datos generados en sus experimentos, para que con ellas queden definidas las condiciones de uso que dicta el autor, algo fundamental ya que cualquier usuario que acceda a la plataforma podrá acceder al repositorio de trazas.
- Se ha implementado la capacidad de permitir a los usuarios enviar trazas al servidor web que no hayan sido generadas utilizando la herramienta diseñada, filtrando aquellas trazas que no sean adecuadas para su publicación. De esta forma se contribuirá con la creación de un gran repositorio de gran diversidad de aplicaciones y configuraciones.

6.2 Trabajos futuros

En esta sección se proponen una serie de ampliaciones o detalles a cambiar para lograr que la plataforma tenga un mayor atractivo para los usuarios del sector de la investigación y diseño de sistemas HPC.

Se propone completar el objetivo que no ha sido posible verificar, la flexibilidad del sistema, implementando un sistema como el propuesto en el apartado 4.3, que permita la posibilidad de añadir distintos sistemas HPC a la plataforma sin necesidad de modificar la funcionalidad referente al servidor web.

Otra propuesta seria mejorar el sistema buscando un proceso de compilación que permita utilizar la herramienta para distintos lenguajes de programación como Fortran, que también es compatible con Extrae, además de C.

Un aspecto muy importante para los usuarios consistiría en añadir una sección de ayuda en la que se explique a los usuarios las distintas peculiaridades de la aplicación, una guía de uso, y otro tipo de información referente a las licencias y a las citas.

Otro cambio de menor importancia en cuanto a funcionalidad se refiere sería la creación un tema visual con el que se presente el contenido del sitio web más limpio y formal.

Por supuesto, la publicación del servidor web será un trabajo a realizar para cumplir los objetivos de la herramienta, y junto a ello será necesario la divulgación de la misma a través de los foros apropiados (conferencias de simulación, encuentros sobre MPI, ...) con el fin de crear una gran comunidad con la que promover la compartición de esta información de gran valor.

BIBLIOGRAFÍA

[1] Bibtex.org, *Your BibTeX resource*. Consulta realizada el 28 de marzo de 2014. [Online]. Available: http://www.bibtex.org/

[2] The CGPOP Miniapp. Consulta realizada el 27 de marzo de 2014. [Online]. Available: http://www.cs.colostate.edu/hpc/cgpop/

[3] Cray, *the Supercoputer Company*. Consulta realizada el 1 de abril de 2014. [Online]. Available: http://www.cray.com/Home.aspx

[4] Drush.org, *A command line shell and scripting interface for Drupal.* Consulta realizada el 15 de noviembre de 2013. [Online]. Available: http://drush.ws/

[5] *Message Passing Interface Forum*, especificaciones del estándar MPI. Conslta realizada el 2 de abril de 2014. [Online]. Available: http://www.mpi-forum.org/

[6] *The Graph 500 List*. Consulta realizada el 27 de marzo de 2014. [Online]. Available: http://www.graph500.org/

[7] Joomla, *The Platform Millions of Websites are built on*. Consulta realizada el 1 de abril de 2014. [Online]. Available: http://www.joomla.org/

[8] Magento, *Ecommerce Software and Ecommerce Platform Solutions*. Consulta realizada el 1 de abril de 2014. [Online]. Available: http://magento.com/

[9] Open MPI, *Open Source High Performance Computing*. Consulta realizada el 2 de noviembre de 2013. [Online]. Available: http://www.open-mpi.org/

[10] Sitio web de CodeMirror. Consulta realizada el 18 de enero de 2014. [Online]. Available: http://codemirror.net/

[11] Sitio web de Creative Commons. Consulta realizada el 13 de marzo de 2014. [Online]. Available: http://creativecommons.org

[12] Sitio web de Drupal. Consulta realizada el 23 de octubre de 2013. [Online]. Available: https://drupal.org

[13] Sitio web de Drupal, API del modulo "views". [Online]. Available: https://api.drupal.org/api/views

[14] Sitio web de Drupal, documentación. Consulta realizada el 20 de noviembre de 2013. [Online]. Available: https://drupal.org/documentation

[15] Sitio web de Drupal, documentación de la API para la generación de formularios. Consulta realizada el 15 de marzo de 2014. [Online]. Available: https://api.drupal.org/api/drupal/includes!form.inc/group/form_api/7

[16] Sitio web de Drupal, documentación del sistema de menús. Consulta realizada el 15 de marzo de 2014. [Online]. Available: https://api.drupal.org/api/drupal/includes!menu.inc/group/menu/7

[17] Sitio web de Drupal, sección de la API (*Application Programming Interface*). Consulta realizada el 25 de noviembre de 2013. [Online]. Available: https://api.drupal.org/api/drupal

[18] Sitio web de la biblioteca de PHP phpseclib. Consulta realizada el 17 de noviembre de 2013. [Online]. Available: http://phpseclib.sourceforge.net/

[19] Sitio web de la compañia SchedMD, documentación de SLURM. Consulta realizada el 24 de febrero de 2014. [Online]. Available: http://www.schedmd.com/slurmdocs/

[20] Sitio web de la herramienta Dyninst. Consulta realizada el 18 de octubre de 2013. [Online]. Available: http://www.dyninst.org/

[21] Sitio web de PAPI (*Performance Application Programming Interface*). Consulta realizada el 15 de octubre de 2013. [Online]. Available: http://icl.cs.utk.edu/papi/index.html

[22] Sitio web de Python. Consulta realizada el 20 de octubre de 2013. [Online]. Available: https://www.python.org/

[23] Sitio web de Python, documentación de la biblioteca ConfigParser. Consulta realizada el 15 de enero de 2014. [Online]. Available: http://docs.python.org/2/library/configparser.html

[24] Sitio web del Centro de Nacional de Supercomputación, BSC, sección de la herramienta Dimemas. Consulta realizada el 16 de octubre de 2013. [Online]. Available: http://www.bsc.es/computer-sciences/performance-tools/dimemas

[25] Sitio web del Centro Nacional de Supercomputación, BSC (*Barcelona Supercomputing Center*). Consulta realizada el 15 de octubre de 2013. [Online]. Available: http://www.bsc.es/

[26] Sitio web del Centro Nacional de Supercomputación, BSC, sección de la herramienta de extracción de trazas Extrae. Consulta realizada el 15 de octubre de 2013. [Online]. Available: http://www.bsc.es/computer-sciences/extrae

[27] Sitio web del Centro Nacional de Supercomputación, BSC, sección de la herramienta de visualización de trazas Paraver. Consulta realizada el 16 de octubre de 2013. [Online]. Available: http://www.bsc.es/computer-sciences/performance-tools/paraver

[28] Sitio web del proyecto GNU, sección de la herramienta GNU Make. Consulta realizada el 2 de noviembre de 2013.

[29] Top 500, clasificación de noviembre de 2013. Consulta realizada el 1 de abril de 2014. [Online]. Available: http://top500.org/lists/2013/11/

[30] Wordpress, emphBlog Tool, Publishing Platform, and CMS. Consulta realizada el 1 de abril de 2014. [Online]. Available: https://wordpress.org/

[31] E. Berdondini. Sitio web de Drupal, página del proyecto Elysia Cron. Consulta realizada el 2 de febrero de 2014. [Online]. Available: https://drupal.org/project/elysia_cron

[32] *Extrae, user guide manual*, Centro Nacional de Supercomputación, BSC, http://www.bsc.es/computer-sciences/performance-tools/trace-generation/extrae/extrae-user-guide.

[33] F. Allen et al, "Blue Gene: A vision for protein science using a petaflop supercomputer," *"IBM Systems Journal"*, vol. 40, pp. 310–327, 2001.

[34] M. Ferran. Sitio web de Drupal, página del proyecto Field Permissions. Consulta realizada el 3 de abril de 2014. [Online]. Available: https://drupal.org/project/field_permissions

[35] Registro de un nuevo tipo de archivo MIME (*Multipurpose Internet Mail Extensions*) denominado .ZIP. IANA (*Internet Assigned Numbers Authority*). 20 de Julio de 1993. [Online]. Available: http://www.iana.org/assignments/media-types/application/zip

[36] L. Hunter. Sitio web de Drupal, guía de intalación de Drupal. Consulta realizada el 10 de noviembre de 2013. [Online]. Available: https://drupal.org/documentation/install

[37] A. Klump. Sitio web de Drupal, página del proyecto XML Field. Consulta realizada el 10 de enero de 2014. [Online]. Available: https://drupal.org/project/xml_field

[38] D. F. Kudwien. Sitio web de Drupal, página del proyecto libraries api. Consulta realizada el 3 de diciembre de 2013. [Online]. Available: https://drupal.org/project/libraries

[39] G. Magini. Sitio web de Drupal, página del proyecto Conditional Fields. Consulta realizada el 10 de enero de 2014. [Online]. Available: https://drupal.org/project/conditional_fields

[40] B. Melançon, J. Luisi, K. Négyesi, G. Anderson, B. Somers, S. Corlosquet, S. Freudenberg, M. Lauer, E. Carlevale, F. Lorétan, D. Nordin, R. Szrama, S. Stewart, J. Strawn, B. Travis, D. Hakimzadeh, A. Scavarda, A. Albala, A. Micka, R. Douglass, R. Monks, R. Scholten, P. Wolanin, K. VanValkenburgh, G. Stout, K. Q. Dolin, F. Mars, S. Boyer, M. Gifford, and C. Sarahe, *The Definite Guide to Drupal 7*, First, Ed. Apress, 2011.

[41] E. Miles. Sito web de Drupal, página del proyecto Views. Consultado el 3 de diciembre de 2013. [Online]. Available: https://drupal.org/project/views

[42] L. Scott. Sitio web de Drupal, documentación sobre la configuración del cron. Consulta realizada el 2 de febrero de 2014. [Online]. Available: https://drupal.org/cron

[43] J. Stals. Sitio web de Drupal, página del proyecto Field Group. Consulta realizada el 14 de enero de 2013. [Online]. Available: https://drupal.org/project/field_group